SONY

SONY COMPUTER ENTERTAINMENT

Cell Broadband Engine Registers

Version 1.0

November 9, 2005

Printed in the United States of America August 2005

Sony Corporation
6-7-35 Kitashinagawa
Shinagawa-ku, Tokyo, 141-0001 Japan

Sony Computer Entertainment Inc.
2-6-21 Minami-Aoyama, Minato-ku,
Tokyo, 107-0062 Japan

The Sony home page can be found at **http://www.sony.net**
The SCEI home page can be found at **http://www.scei.co.jp**

The Cell Broadband Engine home page can be found at **http://cell.scei.co.jp**

November 9, 2005
Version 1.0

Contents

List of Tables

Preface

This document describes the fields of those Cell Broadband Engine (CBE) registers that are used by a wide variety of audiences. This document should be used in conjunction with the *Cell Broadband Engine Architecture* (CBEA) and other supporting documents listed in *Related Publications*.

Who Should Read This Manual

This manual is intended for designers who plan to develop products using the CBE implementation of the CBEA.

Related Publications

A list of related materials follows.

Title	Version	Date
Cell Broadband Engine Architecture	1.0	August 2005
PowerPC User Instruction Set Architecture, Book I	2.02	January 2005
PowerPC Virtual Environment Architecture, Book II	2.02	January 2005
PowerPC Operating Environment Architecture, Book III	2.02	January 2005
Synergistic Processor Unit Instruction Set Architecture	1.0	August 2005

Conventions and Notation

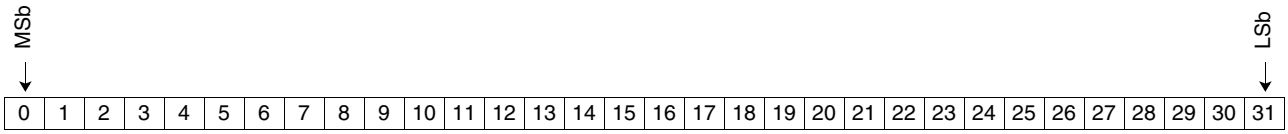
Byte Ordering

Throughout this document, standard IBM big-endian notation is used, meaning that bytes are numbered in ascending order from left to right. Big-endian and little-endian byte ordering are described in the *Cell Broadband Engine Architecture* document.

Note: In this document, storage units are defined as they are defined in the *PowerPC Architecture*. Quadwords are 128 bits, doublewords are 64 bits, words are 32 bits, halfwords are 16 bits, and bytes are 8 bits.

Bit Ordering

Bits are numbered in ascending order from left to right with bit 0 representing the most significant bit (MSb) and bit 31 the least significant bit (LSb).



Bit Encoding

Numbers are generally shown in decimal format, unless designated as follows:

- Hexadecimal values are preceded by an “x” and enclosed in single quotation marks.
  For example: x‘0A00’.
- Binary values in sentences appear in single quotation marks.
  For example: ‘1010’.

The binary point for fixed-point format data is at the right end of the field or value. Operations are performed with the binary points aligned, even if the fields are of different widths.

Software Documentation Conventions

The following software documentation conventions are used in this manual:

- Commands and instruction names are written in **bold** type. For example: **put**.
- Transactions are capitalized and are not bold to distinguish them from instructions. For example: The intent of the enforce in-order execution of I/O (EIEIO) transaction is to act as a barrier for two groups of transactions in support of the PowerPC Architecture **eieio** instruction.
- Variables are written in italic type. Required parameters are enclosed in angle brackets. Optional parameters are enclosed in brackets. For example: **get*<f,b>[s]***.
- I/O signal names are in upper case.

Referencing Registers, Fields, and Bit Ranges

Registers are referenced by their full name or by their short name (also called the register mnemonic). Fields are referenced by their field name or by their bit position. The following table describes how registers, fields, and bit ranges are referenced in this document and provides examples of the references.

Type of Reference	Format	Example
Reference to a specific register and a specific field using the register short name and the field name	Register_Short_Name[Field_Name]	MSR[R]
Reference to a field using the field name	[Field_Name]	[R]
Reference to a specific register and to multiple fields using the register short name and the field names	Register_Short_Name[Field_Name1, Field_Name2]	MSR[FE0, FE1]
Reference to a specific register and to multiple fields using the register short name and the bit positions.	Register_Short_Name[Bit_Number, Bit_Number]	MSR[52, 55]
Reference to a specific register and to a field using the register short name and the bit position or the bit range.	Register_Short_Name[Bit_Number]	MSR[52]
	Register_Short_Name[Starting_Bit_Number:Ending_Bit_Number]	MSR[39:44]
A field name followed by an equal sign (=) and a value indicates the value for that field.	Register_Short_Name[Field_Name]=*n*[1]	MSR[FE0]=1 MSR[FE]=x‘1’
	Register_Short_Name[Bit_Number]=*n*[1]	MSR[52]=0 MSR[52]=x‘0’
	Register_Short_Name[Starting_Bit_Number:Ending_Bit_Number]=*n*[1]	MSR[39:43]=‘10010’ MSR[39:43]=x‘11’

1. Where *n* is the binary or hex value for the field or bits specified in the brackets.

1. Cell Broadband Engine Memory-Mapped I/O Registers

This section defines the memory map for the memory-mapped I/O (MMIO) registers on the Cell Broadband Engine (CBE).

While the *Cell Broadband Engine Architecture* (CBEA) defines one base register (BP_Base, which is known as BE_MMIO_Base in the implementation) for relocating the internal registers, the CBE implements BE_MMIO_Base as several base registers that replicate this relocation function for the units, as shown in *Table 1-1*. These base register values are initialized from the configuration ring during the power-on reset (POR) sequence.

The number of bits in these configuration ring fields is also shown in *Table 1-1*. In all cases, these bits correspond to the most significant bit of the 42-bit real-address implemented in the CBE. The most significant 19 bits of all these configuration ring fields should be set to the same value. If a configuration ring field has more than 19 bits, these additional bits should be set to a value consistent with the settings in *Table 1-3* on page 14 for the starting address of that unit. Each Synergistic Processor Element (SPE) memory flow controller (MFC) unit has its own BE_MMIO_Base in the configuration ring, but each unit should be initialized to the same value. The Input/Output Controller (IOC) unit contains one configuration ring field that defines the most significant 22 bits of the MMIO space for multiple units as shown in *Table 1-1*.

The value of BE_MMIO_Base is relocatable, and the value of the most significant 19 bits is not specified in this document.

Table 1-1. Registers That Are Replicated Forms of BE_MMIO_Base

Configuration Ring Field	Sets BE_MMIO_Base for the Units Below
SPE BE_MMIO_Base Address (19 bits)	SPE0-7
PPE BE_MMIO_Base Address (30 bits)	PPE
MIC CBE_MMIO _Base Address (30 bits)	MIC
PRV BE_MMIO_Base Address (30 bits)	Pervasive
BEI BE_MMIO_Base Address (22 bits)	IIC, IOC Address Translation, IOC,BIC, and EIB

1.1 Classification of Registers

Registers in the MMIO memory map are classified as either Privilege 1, Privilege 2, or Problem State. These designations relate to a suggested hierarchy of privileged access. Privilege 1 registers are the most privileged. They are intended to be accessed by a hypervisor or firmware operating in the HV=1 and PR=0 mode, usually when supporting logical partitioning. Privilege 2 registers are intended for privileged operating system code running in HV=0 and PR=0 mode. When no hypervisor is present, firmware and the privileged operating system typically combine Privilege 1 and Privilege 2 resources into one privilege level. Problem State registers may be directly accessible by applications operating at the HV=0 and PR=1 modes, should an operating system so choose. Access to MMIO registers in these modes is not directly enforced by hardware. Enforcement of these accesses is left to the operating system and hypervisor developers' use of the translation facilities of the PPE memory management unit (MMU) when data relocation is active.

1.2 MMIO Access Rules for 32-Bit and 64-Bit Registers

In general, 32-bit registers are accessed 32 bits at a time and 64-bit registers are accessed 64 bits at a time internally on the CBE. No accesses are allowed on less than 32 bits. The following table lists the access rules for 32-bit and 64-bit registers.

Table 1-2. 32-Bit and 64-Bit Register Access Rules

Address Space	Doubleword Read (bits [0:63])	High Word Read (bits [0:31])	Low Word Read (bits [32:63])	Doubleword Write (bits [0:63])	High Word Write (bits [0:31])	Low Word Write (bits [32:63])
Problem Space	yes	yes	yes	yes	yes	yes
Privilege with high word reserved and low word defined	yes	no	yes	yes	no	yes
Privilege with high word defined and low word reserved	yes	yes	no	yes	yes	no
Privilege space with both high and low word defined	yes	no	no	yes	no	no

1.3 The MMIO Memory Map

Reserved areas of the MMIO Memory Map within assigned units, reserved registers, and reserved bits operate in the same way: writes have no effect and reads return zeros. No error flags are set when reserved locations that are assigned to units are written or read. Reserved areas of the MMIO memory map that are not assigned to any unit should not be read from or written to, as doing so causes serious errors in software.

Table 1-3. CBE Memory Map (Page 1 of 3)

BE_MMIO_Base + Offset Range		Area	Size in Hex	Size in Decimal	Additional Information
Start	End				
x‘0’	x‘03FFFF’	SPE(0) Local Store	x‘40000’	262144	
x‘040000’	x‘05FFFF’	SPE(0) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘060000’	x‘07FFFF’	SPE(0) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘080000’	x‘0BFFFF’	SPE(1) Local Store	x‘40000’	262144	
x‘0C0000’	x‘0DFFFF’	SPE(1) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘0E0000’’	x‘0FFFFF’	SPE(1) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘100000’	x‘13FFFF’	SPE(2) Local Store	x‘40000’	262144	
x‘140000’	x‘15FFFF’	SPE(2) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘160000’	x‘17FFFF’	SPE(2) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘180000’	x‘1BFFFF’	SPE(3) Local Store	x‘40000’	262144	

Table 1-3. CBE Memory Map (Page 2 of 3)

BE_MMIO_Base + Offset Range		Area	Size in Hex	Size in Decimal	Additional Information
Start	End				
x‘1C0000’	x‘1DFFFF’	SPE(3) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘1E0000’	x‘1FFFFF’	SPE(3) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘200000’	x‘23FFFF’	SPE(4) Local Store	x‘40000’	262144	
x‘240000’	x‘25FFFF’	SPE(4) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘260000’	x‘27FFFF’	SPE(4) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘280000’	x‘2BFFFF’	SPE(5) Local Store	x‘40000’	262144	
x‘2C0000’	x‘2DFFFF’	SPE(5) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘2E0000’	x‘2FFFFF’	SPE(5) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘300000’	x‘33FFFF	SPE(6) Local Store	x‘40000’	262144	
x‘340000’	x‘35FFFF’	SPE(6) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘360000’	x‘37FFFF’	SPE(6) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘380000’	x‘3BFFFF’	SPE(7) Local Store	x‘40000’	262144	
x‘3C0000’	x‘3DFFFF’	SPE (7) Problem State	x‘20000’	131072	*Table 3-3 SPE Problem State Memory Map* on page 30
x‘3E0000’	x‘3FFFFF’	SPE(7) Privilege 2	x‘20000’	131072	*Table 3-2 SPE Privilege 2 Memory Map* on page 29
x‘400000’	x‘401FFF’	SPE(0) Privilege1	x‘2000’	8192	*Table 3-1 SPE Privilege 1 Memory Map* on page 26
x‘402000’	x‘403FFF’	SPE(1) Privilege1	x‘2000’	8192	
x‘404000’	x‘405FFF’	SPE(2) Privilege1	x‘2000’	8192	
x‘406000’	x‘407FFF’	SPE(3) Privilege1	x‘2000’	8192	
x‘408000’	x‘409FFF’	SPE(4) Privilege1	x‘2000’	8192	
x‘40A000’	x‘40BFFF’	SPE(5) Privilege1	x‘2000’	8192	
x‘40C000’	x‘40DFFF’	SPE(6) Privilege1	x‘2000’	8192	
x‘40E000’	x‘40FFFF’	SPE(7) Privilege1	x‘2000’	8192	
x‘500000’	x‘500FFF’	PPE Privilege	x‘1000’	4096	*Table 2-1 PPE Privilege MMIO Memory Map* on page 17
x‘501000’	x‘507FFF’	Reserved			
x‘508000’	x‘508FFF’	IIC	x‘1000’	4096	*Table 6-1 IIC Memory Map* on page 101
x‘509000’	x‘5093FF’	Reserved			
x‘509400’	x‘5097FF’	Pervasive: Performance Monitor	x‘400’	1024	*Table 9-1 Pervasive Registers* on page 119
x‘509800’	x‘509BFF’	Pervasive: Thermal and Power Management	x‘400’	1024	*Table 9-1 Pervasive Registers* on page 119

Table 1-3. CBE Memory Map (Page 3 of 3)

BE_MMIO_Base + Offset Range		Area	Size in Hex	Size in Decimal	Additional Information
Start	End				
x‘509C00’	x‘509FFF’	Pervasive: RAS	x‘400’	1024	*Table 9-1* on page 119
x‘50A000’	x‘50AFFF’	MIC and TKM	x‘1000’	4096	*Table 7-1 TKM MMIO Memory Map* on page 111
x‘50B000’	x‘50FFFF’	Reserved			
x‘510000’	x‘510FFF’	IOC Address Translation	x‘1000’	4096	*Table 5-1 IOC Address Translation MMIO Memory Map* on page 89
x‘511000’	x‘5113FF’	BIC 0 NClk	x‘400’	1024	
x‘511400’	x‘5117FF’	BIC 1 NClk	x‘400’	1024	
x‘511800’	x‘511BFF’	EIB	x‘400’	1024	*Table 8-1 EIB MMIO Memory Map* on page 113
x‘511C00’	x‘511FFF’	IOC I/O Command	x‘400’	1024	*Table 4-1 BEI IOC MMIO Memory Map* on page 79
x‘512000’	x‘512FFF’	BIC 0 BClk	x‘1000’	4096	
x‘513000’	x‘513FFF’	BIC 1 BClk	x‘1000’	4096	
x‘514000’	x‘514FFF’	Reserved	x‘1000’	4096	
x‘515000’	x‘7FFFFF’	Reserved			

2. PowerPC Processor Element (PPE) MMIO Registers

This section describes the PPE memory-mapped I/O (MMIO) registers. *Table 2-1* on page 17 shows the PPE MMIO memory map and lists the PPE registers. The PPE register space starts at x‘500 000’ and ends at x‘500 FFF’. Offsets are from the start of the PPE Privilege area. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

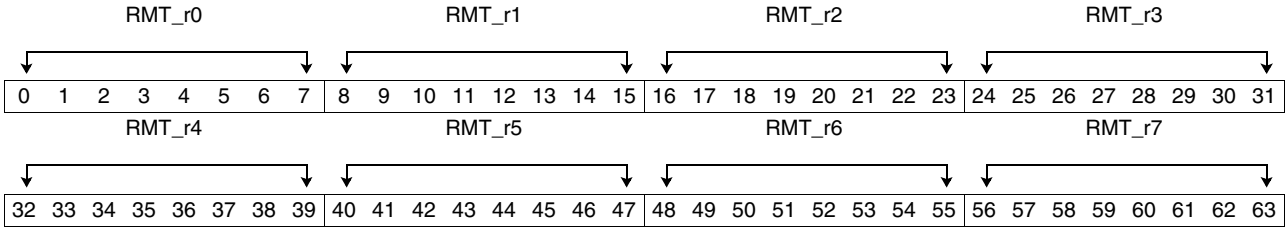
Table 2-1. PPE Privilege MMIO Memory Map

Hexadecimal Offset (x‘500 *nnn*’)	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
Level 2 (L2) Cache MMIO Registers				
x‘300’	*L2 RMT Index Register (L2_RMT_Index)*	64	R/W	Not implemented
x‘310’	*L2 RMT Data Register (L2_RMT_Data)*	64	R/W	*Section 2.1.1* on page 18
x‘800’ – x‘858’	Reserved			
x‘870’	*L2 Machine Check Enable Register (L2_Machchk_en)*	64	R/W	*Section 2.1.2* on page 19
x‘878’	Reserved			
Core Interface Unit (CIU) MMIO Registers				
x‘900’ – x‘930’	Reserved			
x‘938’	*CIU Enable Recoverable Error Register (CIU_ERE)*	64	R/W	*Section 2.2.1* on page 20
x‘940’	*CIU Local Recoverable Error Counter Register (CIU_REC)*	64	R/W	*Section 2.2.2* on page 21
x‘948’	*CIU Mode Setup Register (CIU_ModeSetup)*	64	R/W	*Section 2.2.3* on page 22
x‘958’ – x‘A60’	Reserved			
Bus Interface Unit (BIU) MMIO Registers				
x‘B00’ – x‘B58’	Reserved			
x‘B60’ x‘B68’ x‘B70’	*BIU Reserved Registers 1-3 (BIU_Reserved_n)*	64	R/W	*Section 2.3.1* on page 24

2.1 L2 MMIO Registers

2.1.1 L2 RMT Data Register (L2_RMT_Data)

Register Short Name	L2_RMT_Data	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘500310’	**Memory Map Area**	PPE Privilege
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	L2

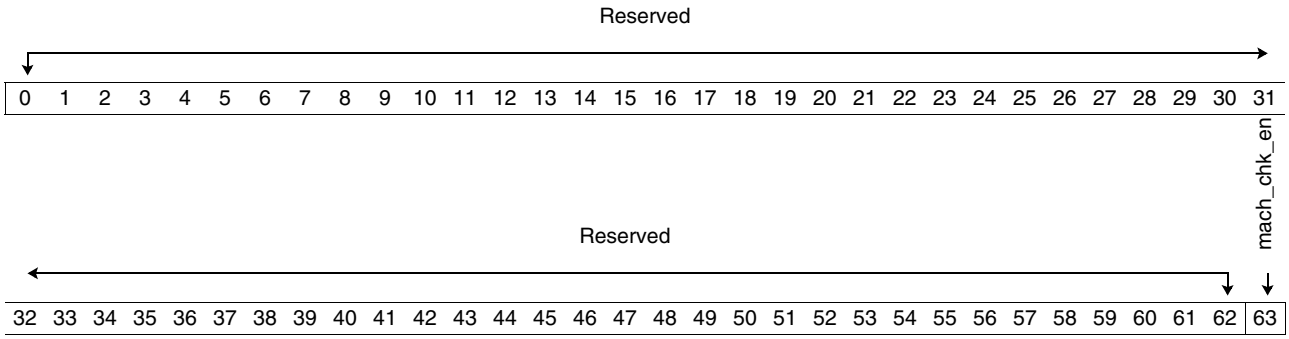


Bit(s)	Field Name	Description
0:7	RMT_r0	Replacement management table (RMT) row 0 (classID = 000)
8:15	RMT_r1	RMT row 1 (classID = 001)
16:23	RMT_r2	RMT row 2 (classID = 010)
24:31	RMT_r3	RMT row 3 (classID = 011)
32:39	RMT_r4	RMT row 4 (classID = 100)
40:47	RMT_r5	RMT row 5 (classID = 101)
48:55	RMT_r6	RMT row 6 (classID = 110)
56:63	RMT_r7	RMT row 7 (classID = 111)

Programming Note: For each RMT row, a field value of ‘00000000’ is treated logically as ‘11111111’. In other words, disabling all sets in an RMT row results in those sets being treated as if they were enabled. This is not allowed.

2.1.2 L2 Machine Check Enable Register (L2_Machchk_en)

Register Short Name	L2_Machchk_en	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'500870'	**Memory Map Area**	PPE Privilege
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	L2



Bit(s)	Field Name	Description
0:62	Reserved	Bits are not implemented; all bits read back zero.
63	mach_chk_en	machine_chk_en_lt 0 Disable v_t_pu_machchk signal from going active and allow v_t_rec_err to go active due to a machine check. 1 Allow v_t_pu_machchk signal to go active and disable v_t_rec_err from going active due to a machine check. v_t_pu_machchk and v_t_rec_err are also gated with FIR(3) Checkstop_en and Mask bits. machine_chk_en_lt does not mask the v_q_rld_machine_chk signal on the reload bus, which is sent to the PPU via the L2 or CIU.

2.2 CIU MMIO Registers

2.2.1 CIU Enable Recoverable Error Register (CIU_ERE)

Register Short Name	CIU_ERE	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘500938’	**Memory Map Area**	PPE Privilege
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	CIU

Bit(s)	Field Name	Description
0	IP	Enable a recoverable instruction cache parity error to increment the local recoverable-error counter.
1	DP	Enable a recoverable data cache parity error to increment the local recoverable-error counter.
2:63	Reserved	Bits are not implemented; all bits read back zero.

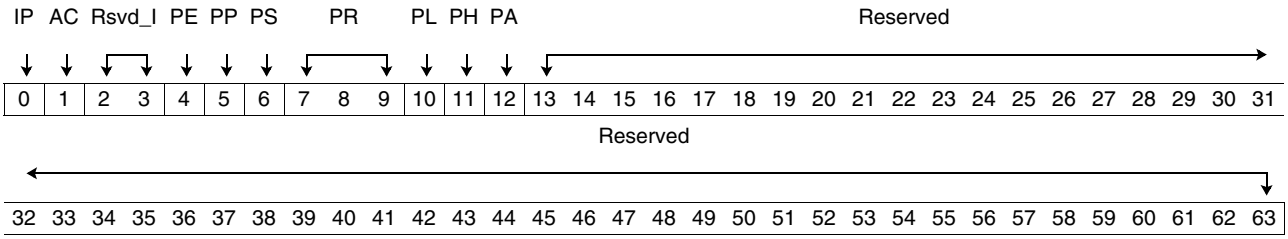
2.2.2 CIU Local Recoverable Error Counter Register (CIU_REC)

Register Short Name	CIU_REC	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘500940’	**Memory Map Area**	PPE Privilege
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	CIU

Bit(s)	Field Name	Description
0:5	RC	Local recoverable-error counter for the PPU Counts the PPU’s recoverable errors (L1 Instruction Cache parity errors and L1 Data Cache parity errors) until the counter reaches its maximum value (all ones). At that time, the local carry signal is asserted for the test control unit (TCU). The error signals are converted to NClk/2 signals and ORed so they can be counted. Thus, there can be a miscount. If there is a checkstop error, the counter freezes. The TCU can reset this counter to zero. The register counts up to 63 errors and freezes there until reset by the TCU or preset via MMIO.
6:63	Reserved	Bits are not implemented; all bits read back zero.

2.2.3 CIU Mode Setup Register (CIU_ModeSetup)

Register Short Name	CIU_ModeSetup	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘500948’	**Memory Map Area**	PPE Privilege
Value at Initial POR	x‘01000000_00000000’	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	CIU



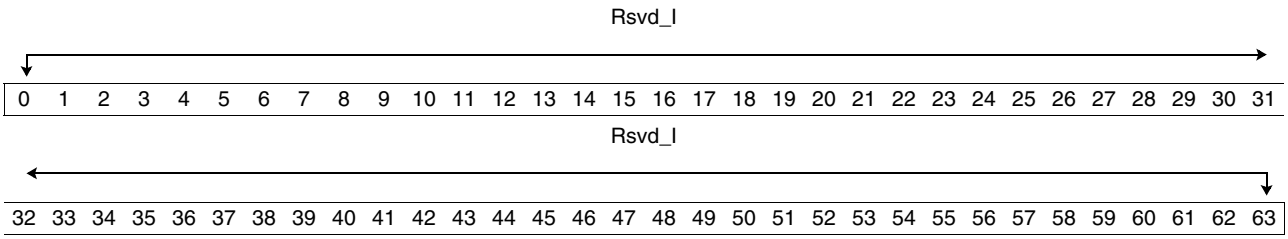
Bit(s)	Field Name	Description
0	IP	CIU Load Queue (CLQ) Instruction High-Priority Mode. The CIU arbiter normally gives a higher priority to data load requests. 0 Both threads on the processor run in caching-inhibited space when both threads issue caching-inhibited instruction fetches. 1 The arbiter grants instruction fetches higher priority than data loads.
1	AC	CLQ Always-Correct Mode When this bit is set, the CIU forces active the always-correct signal to the L2 for each request, which puts the L2 into Always-Correct mode. 0 Pass to L2 from requester 1 Force active to L2
2:3	Rsvd_I	Reserved. Latch bits are implemented; value read is the value written.
4	PE	Data Prefetch Enable 0 Treat data prefetch as a no-op. 1 Enable data prefetch To disable data prefetch, deactivate all data prefetch streams before resetting this bit.
5	PP	Data Prefetch 4-KB Page Size 0 Set the read size (0 to 1) from the PPE. 1 Limit data prefetch size to a maximum of 4 KB and stop at the 4-KB boundary.
6	PS	Data Prefetch 4/8 Disable The prefetch queue has eight entries to support the maximum number of streams. The entries are divided to support two threads. When both threads are running, each gets four prefetches. If only one thread is running, it gets eight prefetches (both halves). When this bit is set, only four entries are allocated per thread, regardless of the number of active threads. 0 Eight entries for a single thread 1 Four entries for a single thread
7:9	PR	Prefetch Outstanding Request Limit [0:2] This sets the limit for how many outstanding instruction or data prefetch requests can be in the L2. It sets the maximum numbers of L2 RC state machine resources for instruction or data prefetch. The default is ‘100’ (4). If the Prefetch Outstanding Request Limit 4 is set (CIU_ModeSetup[10] = `1’), then setting this bit has no effect. The values from ‘001’ (1) to ‘110’ (6) are valid for this limit number. In Data Prefetch Permanent High-Priority Mode (CIU_ModeSetup[11] = `1’), this limit value must be less than ‘110’ (6).

Bit(s)	Field Name	Description
10	PL	Prefetch Outstanding Request Limit 4 0 Enables the Prefetch Outstanding Request Limit. 1 Disables the Prefetch Outstanding Request Limit and forces the count to four.
11	PH	Data Prefetch Permanent High-Priority Mode If this bit is set, the priorities of load-type requests for L2 are changed to give higher priority to the data prefetch request. 0 Data prefetch low-priority mode: MMU > Demand Load > Demand Fetch > Instruction Prefetch > Data Prefetch. If the instruction high-priority mode is set, Demand Load and Demand Fetch are swapped. 1 Data prefetch high-priority mode: MMU > Data Prefetch > Demand Load > Demand Fetch > Instruction Prefetch. If the instruction high-priority mode is set, Demand Load and Demand Fetch are swapped. In the Data Prefetch Permanent High-Priority Mode, the Prefetch Outstanding Request Limit must be less than '110' (6).
12	PA	Data Prefetch Periodical High-Priority Mode This bit has no effect if PH (CIU_ModeSetup[11] = `1') is set. If the PA (CIU_ModeSetup[12] = `1') is set, the priorities of load-type requests for L2 are changed to give higher priority to the data prefetch request. This only occurs after the CIU issues 31 complete load-type requests to the L2 (except for the data prefetch from the last completed request of data prefetch) and the L2 acknowledges these requests. The changed priorities for PA (CIU_ModeSetup[12] = `1') are as follows: MMU > Data Prefetch > Demand Load > Demand Fetch > Instruction Prefetch
13:63	Reserved	Bits are not implemented; all bits read back zero.

2.3 BIU MMIO Registers

2.3.1 BIU Reserved Registers 1-3 (BIU_Reserved_n)

Register Short Name	BIU_Reserved_1 BIU_Reserved_2 BIU_Reserved_3	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘500B60’ x‘500B68’ x‘500B70’	**Memory Map Area**	PPE Privilege
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	BIU



Bit(s)	Field Name	Description
0:63	Rsvd_I	Reserved. Latch bits are implemented; value read is the value written.

3. Synergistic Processor Element (SPE) MMIO Registers

This section describes the SPE memory-mapped I/O (MMIO) registers. Registers and areas marked as implementation specific are not part of the architecture. The detailed description of each register includes the complete hex offset from BE_MMIO_Base. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

For a list of SPE memory-mapped control registers, see:

- *Section 3.1.1 SPE Privilege 1 Memory Map*
- *Section 3.1.2 SPE Privilege 2 Memory Map*
- *Section 3.1.3 SPE Problem State Memory Map*

Notes on the register definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

3.1 SPE Memory Map and Summary Tables

The detailed description of each register includes the complete hex offset from BE_MMIO_Base. The summary tables below list the registers by memory map area, so only the last four digits of the offset are included.

3.1.1 SPE Privilege 1 Memory Map

Table 3-1. SPE Privilege 1 Memory Map (Page 1 of 3)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
MFC Registers *(CBEA Architected Registers)*				
x‘0000’	*MFC State Register 1 (MFC_SR1)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0008’	Reserved			
x‘0010’	*SPU Identification Register (SPU_ID)*	64	R	*Section 3.2.1.1* on page 32
x‘0018’	*MFC Version Register (MFC_VR)*	64	R	See *Appendix A Registers Defined in the CBEA*
x‘0020’	*SPU Version Register (SPU_VR)*	64	R	See *Appendix A Registers Defined in the CBEA*
x‘0028’ – x‘00FF’	Reserved			
Interrupt Registers (*CBEA Architected Registers)*				
x‘0100’	*Class 0 Interrupt Mask Register (INT_Mask_class0)*	64	R/W	*Section 3.2.2.1* on page 33
x‘0108’	*Class 1 Interrupt Mask Register (INT_Mask_class1)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0110’	*Class 2 Interrupt Mask Register (INT_Mask_class2)*	64	R/W	*Section 3.2.2.2* on page 34
x‘0118’ – x‘013F’	Reserved			
x‘0140’	*Class 0 Interrupt Status Register (INT_Stat_class0)*	64	R/W	*Section 3.2.2.3* on page 35
x‘0148’	*Class 1 Interrupt Status Register (INT_Stat_class1)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0150’	*Class 2 Interrupt Status Register (INT_Stat_class2)*	64	R/W	*Section 3.2.2.4* on page 36
x‘0158’ – x‘01FF’	Reserved			
Atomic Unit Control Registers *(Implementation-Specific Registers)*				
x‘0200’	*MFC Atomic Flush Register (MFC_Atomic_Flush)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0280’ – x‘03F8’	Reserved			
MFC TLB Management Registers *(CBEA Architected Registers)*				
x‘0400’	*MFC Storage Description Register (MFC_SDR)*	64	R/W	*Section 3.2.3.1* on page 37
x‘0408’ – x‘04FF’	Reserved			
x‘0500’	*MFC TLB Index Hint Register (MFC_TLB_Index_Hint)*	64	R	*Section 3.2.3.2* on page 38
x‘0508’	*MFC TLB Index Register (MFC_TLB_Index)*	64	R/W	*Section 3.2.3.3* on page 39

Table 3-1. SPE Privilege 1 Memory Map (Page 2 of 3)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
x‘0510’	*MFC TLB Virtual-Page Number Register (MFC_TLB_VPN)*	64	R/W	*Section 3.2.3.4* on page 40
x‘0518’	*MFC TLB Real Page Number Register (MFC_TLB_RPN)*	64	R/W	*Section 3.2.3.5* on page 41
x‘0520’ – x‘053F’	Reserved			
x‘0540’	*MFC TLB Invalidate Entry Register (MFC_TLB_Invalidate_Entry)*	64	W	*Section 3.2.3.6* on page 43
—	*MFC TLB Invalidate All Register (MFC_TLB_Invalidate_All)*	—	—	Not implemented
x‘0580’	Reserved			
MFC Status and Control Registers *(CBEA Architected Registers)*				
x‘0600’	*MFC Address Compare Control Register (MFC_ACCR)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0610’	*MFC Data-Storage Interrupt Status Register (MFC_DSISR)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0620’	*MFC Data Address Register (MFC_DAR)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0628’ – x‘06FF’	Reserved			
Replacement Management Table (RMT) Registers *(Implementation-Specific Registers)*				
—	*MFC TLB Replacement Management Table Index Register (MFC_TLB_RMT_Index)*	—	—	Not implemented
x‘0710’	*MFC TLB Replacement Management Table Data Register (MFC_TLB_RMT_Data)*	64	R/W	*Section 3.2.4* on page 45
x‘0718’ – x‘07FF’	*SPU_RMT_ImplRegs*	—	—	Not implemented
MFC Command Data-Storage Interrupt Registers *(Implementation-Specific Registers)*				
x‘0800’	*MFC Data-Storage Interrupt Pointer Register (MFC_DSIPR)*	64	R	*Section 3.2.5.1* on page 46
x‘0808’	*MFC Local Store Address Compare Register (MFC_LSACR)*	64	R/W	*Section 3.2.5.2* on page 47
x‘0810’	*MFC Local Store Compare Results Register (MFC_LSCRR)*	64	R	*Section 3.2.5.3* on page 48
x‘0818’	Reserved			
x‘0820’	*MFC Transfer Class ID Register (MFC_TClassID)*	64	R/W	*Section 3.2.5.4* on page 49
x‘0828’ – x‘08FF’	Reserved			
Real-Mode Support Registers *(CBEA Architected Register)*				
x‘0900’	*MFC Real Mode Address Boundary Register (MFC_RMAB)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘0908’ – x‘0BFF’	Reserved			
MFC Command Error Register				
x‘0C00’	*MFC Command Error Register (MFC_CER)*	64	R	*Section 3.2.6.1* on page 51
x‘0C08’ – x‘13FF’	Reserved			

Table 3-1. SPE Privilege 1 Memory Map (Page 3 of 3)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
Performance Monitor Register *(Implementation-Specific Register)*				
x‘1400’	*Performance Monitor/Trace Tag Status Wait Mask Register (PM_Trace_Tag_Wait_Mask)*	64	R/W	*Section 3.2.7.1* on page 52
x‘1408’ – x‘1FFF’	Reserved			

3.1.2 SPE Privilege 2 Memory Map

Multiple address offsets for a register indicate that there are multiple instances of this register.

Table 3-2. SPE Privilege 2 Memory Map (Page 1 of 2)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
MFC Registers				
x‘0000’ – x‘10FF’	Reserved			
SLB Management Registers *(CBEA Architected Registers)*				
x‘1100’	Reserved			
x‘1108’	*SLB Index Register (SLB_Index)*	64	R/W	*Section 3.3.1.1* on page 53
x‘1110’	*SLB Effective Segment ID Register (SLB_ESID)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘1118’	*SLB Virtual Segment ID Register (SLB_VSID)*	64	R/W	*Section 3.3.1.2* on page 54
x‘1120’	*SLB Invalidate Entry Register (SLB_Invalidate_Entry)*	64	W	*Section 3.3.1.3* on page 55
x‘1128’	*SLB Invalidate All Register (SLB_Invalidate_All)*	64	W	See *Appendix A Registers Defined in the CBEA*
x‘1130’ – x‘1FFF’	Reserved			
Context Save/Restore Register *(Implementation-Specific Register)*				
x‘2000’ – x‘22FF’	*MFC Command Queue Context Save/Restore Register (MFC_CQ_SR)*	64	W	*Section 3.3.2.1* on page 56
x‘2300’ – x‘2FF8’	Reserved			
MFC Control Register *(CBEA Architected Register)*				
x‘3000’	*MFC Control Register (MFC_CNTL)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘3008’ – x‘3FFF’	*MFC_Cntl1_ImplRegs*	—	—	Not implemented
Interrupt Mailbox Register *(Implementation-Specific Register)*				
x‘4000’	*SPU Outbound Interrupt Mailbox Register (SPU_Out_Intr_Mbox)*	64	R	See *Appendix A Registers Defined in the CBEA*
SPU Control Registers *(CBEA Architected Registers)*				
x‘4040’	*SPU Privileged Control Register (SPU_PrivCntl)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘4058’	*SPU Local Store Limit Register (SPU_LSLR)*	64	R/W	*Section 3.3.2.2* on page 61
x‘4060’	*SPU Channel Index Register (SPU_ChnlIndex)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘4068’	*SPU Channel Count Register (SPU_ChnlCnt)*	64	R/W	*Section 3.3.2.3* on page 62
x‘4070’	*SPU Channel Data Register (SPU_ChnlData)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘4078’	*SPU Configuration Register (SPU_Cfg)*	64	R/W	See *Appendix A Registers Defined in the CBEA*
x‘4080’ – x‘4FFF	Reserved			
Context Save and Restore Registers *(Implementation-Specific Registers)*				
x‘5000’	Reserved			

Table 3-2. SPE Privilege 2 Memory Map (Page 2 of 2)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
x‘5008’	*Context Save and Restore for SPU MFC Commands Register (MFC_CSR_TSQ)*	64	R/W	*Section 3.3.3.1* on page 64
x‘5010’	*Context Save and Restore for SPU MFC Commands Register (MFC_CSR_CMD1)*	64	R/W	*Section 3.3.3.2* on page 65
x‘5018’	*Context Save and Restore for SPU MFC Commands Register (MFC_CSR_CMD2)*	64	R/W	*Section 3.3.3.3* on page 66
x‘5020’	*Context Save and Restore for SPU Atomic Immediate Command (MFC_CSR_ATO)*	64	R/W	*Section 3.3.3.4* on page 67
x‘5028’ – x‘1FFFF	Reserved			

3.1.3 SPE Problem State Memory Map

Multiple address offsets for a register indicate that there are multiple instances of this register.

Table 3-3. SPE Problem State Memory Map (Page 1 of 2)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
SPE Multisource Synchronization Register *(Implementation-Specific Register)*				
x‘0000’	*MFC Multisource Synchronization Register (MFC_MSSync)*	64	R/W	*Section 3.4.1.1* on page 68
x‘0008’ – x‘2FF8’	Reserved			
MFC Command Parameter Registers *(CBEA Architected Registers)*				
x‘3000’	Reserved			
x‘3004’	*MFC Local Store Address Register (MFC_LSA)*	32	W	*Section 3.4.2.1* on page 69
x‘3008’	*MFC Effective Address High Register (MFC_EAH)*	32	R/W	*Section 3.4.2.2* on page 69
x‘300C’	*MFC System Memory Address Register (MFC_EAL)*	32	R/W	See *Appendix A Registers Defined in the CBEA*
x‘3010’	*MFC Transfer Size Register (MFC_Size)*	16	R/W	See *Appendix A Registers Defined in the CBEA*
	MFC Command Tag Register (MFC_Tag)	16	R/W	See *Appendix A Registers Defined in the CBEA*
x‘3014’	*MFC Class ID Command Opcode Register (MFC_ClassID_CMD)*	32	W	*Section 3.4.2.3* on page 70
x‘3014’	*MFC Command Status Register (MFC_CMDStatus)*	32	R	*Section 3.4.2.4* on page 71
Reserved Area				
x‘3020’ – x‘30FF’	Reserved			
MFC Command Queue Control Registers *(CBEA Architected Registers)*				
x‘3104’	*MFC Queue Status Register (MFC_QStatus)*	32	R	*Section 3.4.3.1* on page 72
x‘3204’	*Proxy Tag-Group Query Type Register (Prxy_QueryType)*	32	R/W	See *Appendix A Registers Defined in the CBEA*
x‘321C’	*Proxy Tag-Group Query Mask Register (Prxy_QueryMask)*	32	R/W	See *Appendix A Registers Defined in the CBEA*

Table 3-3. SPE Problem State Memory Map (Page 2 of 2)

Hexadecimal Offset	Register Name and (Short Name)	Width	Read/ Write	Additional Information
x‘322C’	*Proxy Tag-Group Status Register (Prxy_TagStatus)*	32	R	See *Appendix A Registers Defined in the CBEA*
Reserved Area				
x‘3330’ – x‘3FFF’	Reserved			
SPU Control Registers *(CBEA Architected Registers)*				
x‘4004’	*SPU Outbound Mailbox Register (SPU_Out_Mbox)*	32	R	See *Appendix A Registers Defined in the CBEA*
x‘400C’	*SPU Inbound Mailbox Register (SPU_In_Mbox)*	32	W	See *Appendix A Registers Defined in the CBEA*
x‘4014’	*SPU Mailbox Status Register (SPU_Mbox_Stat)*	32	R	*Section 3.4.3.2* on page 73
x‘401C’	*SPU Run Control Register (SPU_RunCntl)*	32	R/W	*Section 3.4.3.3* on page 74
x‘4024’	*SPU Status Register (SPU_Status)*	32	R	*Section 3.4.3.4* on page 75
x‘4034’	*SPU Next Program Counter Register (SPU_NPC)*	32	R/W	*Section 3.4.3.5* on page 77
Reserved Area				
x‘4038’ – x‘13FFF’	Reserved			
Signal Notification Registers *(CBEA Architected registers)*				
x‘1400C’	*SPU Signal Notification Register 1 (SPU_Sig_Notify_1)*	32	R/W	See *Appendix A Registers Defined in the CBEA*
x‘14010’ – x‘1BFFF’	Reserved			
x‘1C00C’	*SPU Signal Notification Register 2 (SPU_Sig_Notify_2)*	32	R/W	See *Appendix A Registers Defined in the CBEA*
x‘1C010’ – x‘1FFFF’	Reserved			

3.2 SPE Privilege 1 Memory Map Registers

This section lists the registers included in the SPE Privilege 1 memory map, with the exception of those that are defined in the CBEA. See *Appendix A Registers Defined in the CBEA* for information on those registers.

3.2.1 MFC Register

This area includes the SPU ID register.

3.2.1.1 SPU Identification Register (SPU_ID)

The SPU_ID register is a read-only register used to distinguish a particular SPU from other SPUs in the system. This register is accessible from the PPE using a load doubeword (**ld**) instruction. Read access to the SPU_ID register from the PPE is privileged, and access from the SPU to this register is not provided. There is one SPU_ID register for each SPU in the Cell Broadband Engine.

For additional information, see the *Cell Broadband Engine Architecture* document.

Register Short Name	SPU_ID	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400010’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	x‘00000000_000000NU’ (N = Node_ID, U = Unit_ID)	**Value During POR Set By**	Bits 56-59 set by configuration ring Bits 60-63 hardwired per SPE on chip
Specification Type	CBEA architected register	**Unit**	MFC

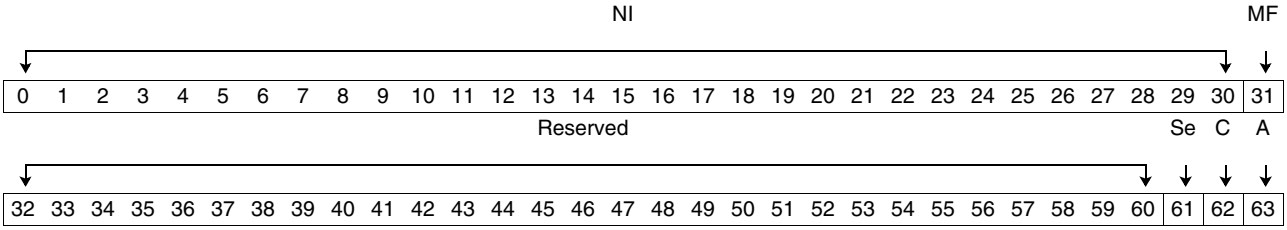
Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32:55	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
56:59	Node_ID	A 4-bit number that identifies the Node ID. This is set in the configuration ring, SPU/SBI[204:207] per SPE on chip.
60:63	Unit_ID	A 4-bit hardwired number that identifies the Unit ID in the bus

3.2.2 Interrupt Registers

There are interrupt mask and interrupt status registers in each MFC: one for each class of interrupt (error, translation, application). The interrupt registers allow privileged software to select which MFC and SPU events are allowed to generate an external interrupt to the PPE. Each bit of the mask registers has a corresponding status bit. The mask register determines which bits are reported in the status register. See the *Cell Broadband Engine Architecture* document for more information about these registers.

3.2.2.1 Class 0 Interrupt Mask Register (INT_Mask_class0)

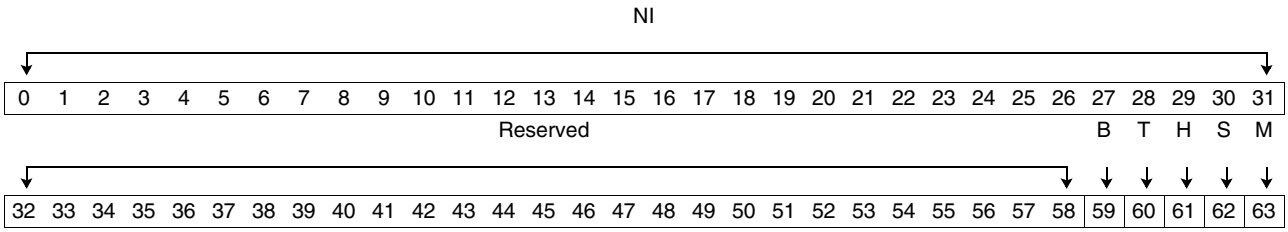
Register Short Name	INT_Mask_class0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400100’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:30	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
31	MF	Enable for MFC_FIR interrupt 0 Interrupt disabled 1 Interrupt enabled
32:60	Reserved	Bits are not implemented; all bits read back zero.
61	Se	Enable for SPU error interrupt 0 Interrupt disabled 1 Interrupt enabled
62	C	Enable for invalid DMA command interrupt 0 Interrupt disabled 1 Interrupt enabled
63	A	Enable for MFC DMA alignment interrupt 0 Interrupt disabled 1 Interrupt enabled

3.2.2.2 Class 2 Interrupt Mask Register (INT_Mask_class2)

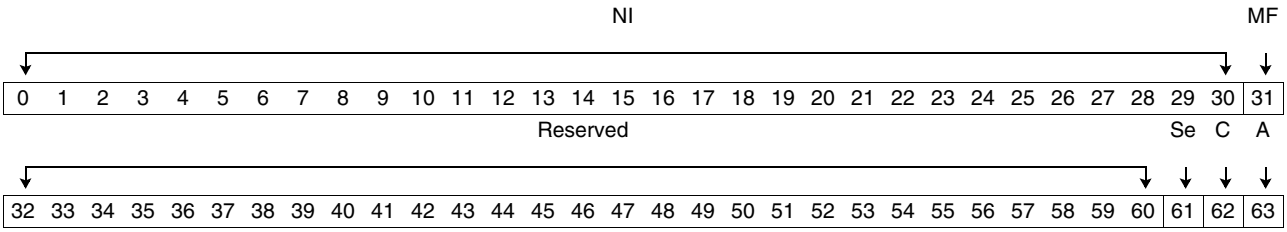
Register Short Name	INT_Mask_class2	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400110’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:31	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
32:58	Reserved	Bits are not implemented; all bits read back zero.
59	B	Enable for SPU Mailbox threshold interrupt 0 Interrupt disabled 1 Interrupt enabled
60	T	Enable for Tag group completion 0 Interrupt disabled 1 Interrupt enabled
61	H	Enable for SPU Halt instruction trap 0 Interrupt disabled 1 Interrupt enabled
62	S	Enable for SPU Stop-and-signal instruction trap 0 Interrupt disabled 1 Interrupt enabled
63	M	Enable for Mailbox interrupt 0 Interrupt disabled 1 Interrupt enabled

3.2.2.3 Class 0 Interrupt Status Register (INT_Stat_class0)

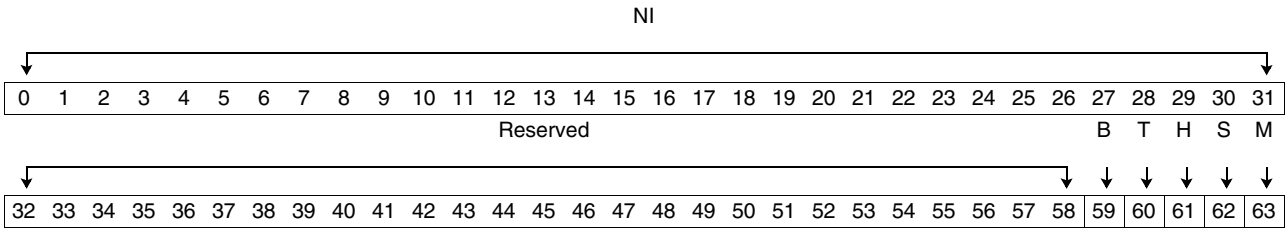
Register Short Name	INT_Stat_class0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400140’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:30	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
31	MF	Status for MFC_FIR interrupt. The invalid DMA command interrupt bit [62], DMA alignment interrupt bit [63], and five hang livelock indication conditions defined in the MFC_FIR do not generate an INT_Stat_class0[31] class 0 interrupt. 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type
32:60	Reserved	All bits read back zero.
61	Se	Status for SPU error interrupt. If there are invalid SPU instructions in SPU_Status[26] and if these interrupts are enabled in SPU_ERR_Mask[63], then this bit (INT_Stat_class0[61] goes to ‘1’. Also for uncorrectable ECC errors. Status is in SPU_ECC_Stat[62]; control is in SPU_ECC_Cntl[62]. 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type **Note:** This implementation differs from the *Cell Broadband Engine Architecture*.
62	C	Status for invalid DMA command interrupt 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type
63	A	Status for DMA alignment interrupt 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type

3.2.2.4 Class 2 Interrupt Status Register (INT_Stat_class2)

Register Short Name	INT_Stat_class2	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400150’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:31	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
32:58	Reserved	All bits read back zero.
59	B	Status for SPU Mailbox threshold interrupt 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type
60	T	Status for DMA Tag group complete interrupt 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type
61	H	Status for SPU Halt instruction trap 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type **Note:** This implementation differs from the *Cell Broadband Engine Architecture*.
62	S	Status for SPU Stop-and-signal instruction trap 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type.
63	M	Status for Mailbox interrupt 0 Interrupt not pending for corresponding interrupt type 1 Interrupt pending for corresponding interrupt type

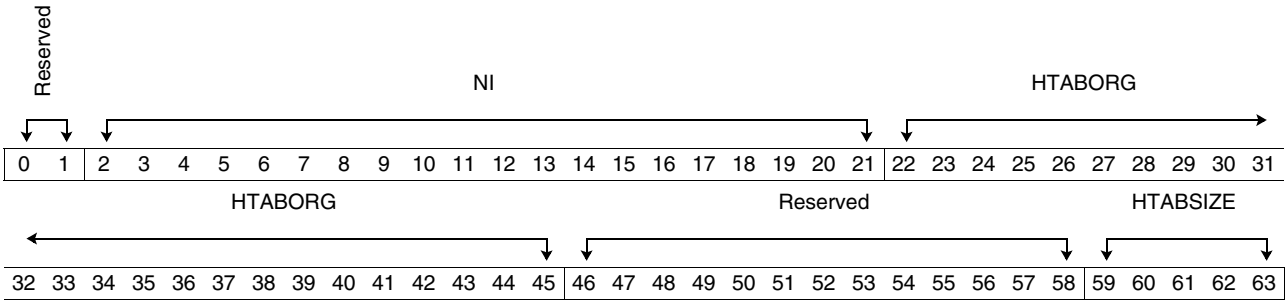
3.2.3 MFC TLB Management Registers

These registers support software TLB management.

3.2.3.1 MFC Storage Description Register (MFC_SDR)

The MFC_SDR register contains the hash table origin and size. The functionality is identical to the PPE SDR1 register (see the *PowerPC Architecture, Book III* for more information). The implemented bits are shown below.

Register Short Name	MFC_SDR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400400’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



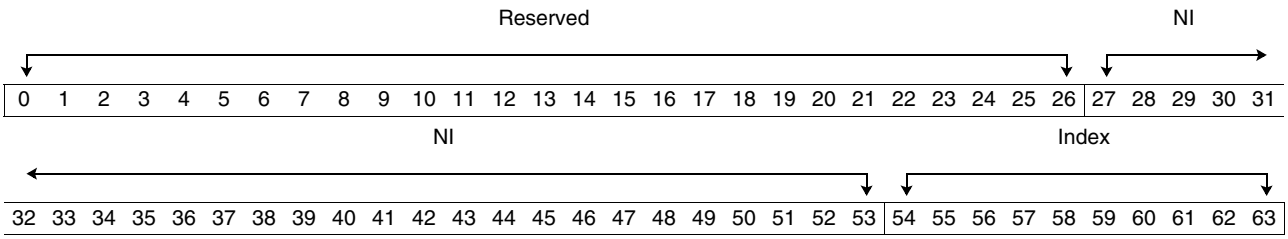
Bit(s)	Field Name	Description
0:1	Reserved	Bits are not implemented; all bits read back zero.
2:21	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
22:45	HTABORG	Page-table origin (real address of the page table). The HTABORG field contains the high-order 44 bits of the 62-bit real address of the page table. The page table is thus constrained to lie on a minimum 2^{18}-byte (256 KB) boundary. The number of low-order zero bits in HTABORG must be greater than or equal to the value in HTABSIZE.
46:58	Reserved	Bits are not implemented; all bits read back zero.
59:63	HTABSIZE	Encoded size of page table. The HTABSIZE field contains an integer giving the number of bits (in addition to the minimum of 11 bits) from the hash that are used in the page-table index. This number must not exceed 28.

3.2.3.2 MFC TLB Index Hint Register (MFC_TLB_Index_Hint)

This register contains the MFC Translation Lookaside Buffer (TLB) Index and congruence class for the most likely candidate for replacement when the SMM has a translation miss in the TLB. The index is written for both hardware and software tablewalks. Software may use this index as a suggestion or make an index of its own.

The CBE uses the lower 12 bits of this register to index the TLB cache. For the PPE_TLB_Index_Hint register, bits [52:59] contain the TLB congruence class, and bits [60:63] contain the set within the congruence class. For the SPE (MFC_TLB_Index_Hint register), bits [54:59] contain the TLB congruence class, and bits [60:63] contain the set within the congruence class.

Register Short Name	MFC_TLB_Index_Hint	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400500’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM



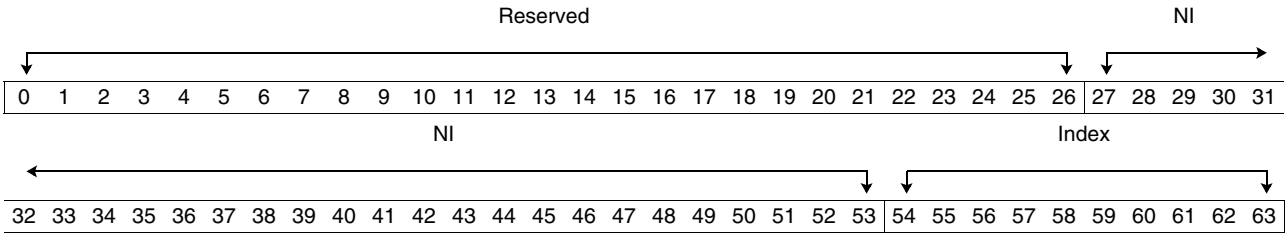
Bit(s)	Field Name	Description
0:26	Reserved	Bits are not implemented; all bits read back zero.
27:53	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
54:63	Index	[54:59] Selects one of the 64 congruence classes [60:63] Selects one of the four entries in the congruence class. These 4 bits decide which entry of the congruence class is updated after a tablewalk. After a tablewalk completes and new data from a PTE is ready to be written into the TLB, these bits indicate the array selected. The SMM hardware writes this register any time there is a TLB miss. Since hardware writes this register directly, only valid combinations are possible. 0000 Does not update a TLB array; causes another tablewalk the next time that address is translated 0001 Selects entry 3 (TLB array 3, the fourth of four entries in the congruency class) 0010 Selects entry 2 (TLB array 2) 0100 Selects entry 1 (TLB array 1) 1000 Selects entry 0 (TLB array 0) All other combinations are not valid. Asserting more than one bit writes more than 1 array and corrupts the TLB data, causing a multiple hit on the next address translation.

3.2.3.3 MFC TLB Index Register (MFC_TLB_Index)

The MFC_TLB_Index register points to a TLB entry (virtual page number (VPN) and real page number (RPN)) to be read or written by MMIO. This register must be written before any access to the TLB array.

The CBE uses the lower 12 bits of the TLB Index Register to index the TLB cache. For the PPE_TLB_Index register, bits [52:59] contain the TLB congruence class, and bits [60:63] contain the set within the congruence class. For the SPE (MFC_TLB_Index), bits [54:59] contain the TLB congruence class, and bits [60:63] contain the set within the congruence class.

Register Short Name	MFC_TLB_Index	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400508’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM



Bit(s)	Field Name	Description
0:15	Reserved	Bits are not implemented; all bits read back zero.
16:20	LVPN	These lower virtual-page number (LVPN) bits [16:20] correspond to the virtual address bits [63:67]. This field is updated during a VPN read. When writing a TLB entry via MMIO, the LVPN data is concatenated with the abbreviated virtual page number (AVPN) data from the MFC_TLB_VPN register. The LVPN data is the least significant portion of the abbreviated page index in the implemented AVPN field.
21:26	NI	This portion of the LVPN field is not implemented in the CBE, but these bits are defined in the *Cell Broadband Engine Architecture*. All bits read back zero.
27:53	NI	This portion of the Index field is not implemented in the CBE, but these bits are defined in the *Cell Broadband Engine Architecture*. All bits read back zero.
54:63	Index	[54:59] Selects one of the 64 congruency classes [60:63] Selects one of the four entries in the congruency class. (One-hot decoding. See MFC_TLB_Index_Hint[54:63] for more information.)

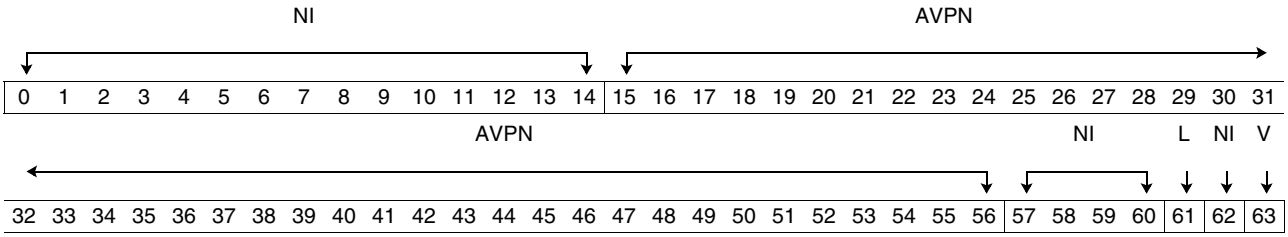
3.2.3.4 MFC TLB Virtual-Page Number Register (MFC_TLB_VPN)

The MFC_TLB_VPN register contains the virtual-page number used in translating the effective address to a real address.

To write a new entry in the TLB via MMIO, the MFC_TLB_Index register should be written first. The index points to the target congruence class and entry. The RPN data should be written after the index, followed by a write to the VPN register. RPN and VPN write data is collected, and then the entire TLB entry is written at once. Any write to the VPN register sets off a write to the TLB array, whether the Index and RPN were previously written or not.

To read the MFC_TLB_VPN data, first write the MFC_TLB_Index to specify the entry. The VPN read command retrieves data from the TLB array entry specified by the MFC_TLB_Index value.

Register Short Name	MFC_TLB_VPN	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400510’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM



Bit(s)	Field Name	Description
0:14	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
15:56	AVPN	Abbreviated virtual-page number
57:60	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
61	L	Large page 0 4-KB page 1 Large page
62	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
63	V	Valid bit 0 Invalid 1 Valid

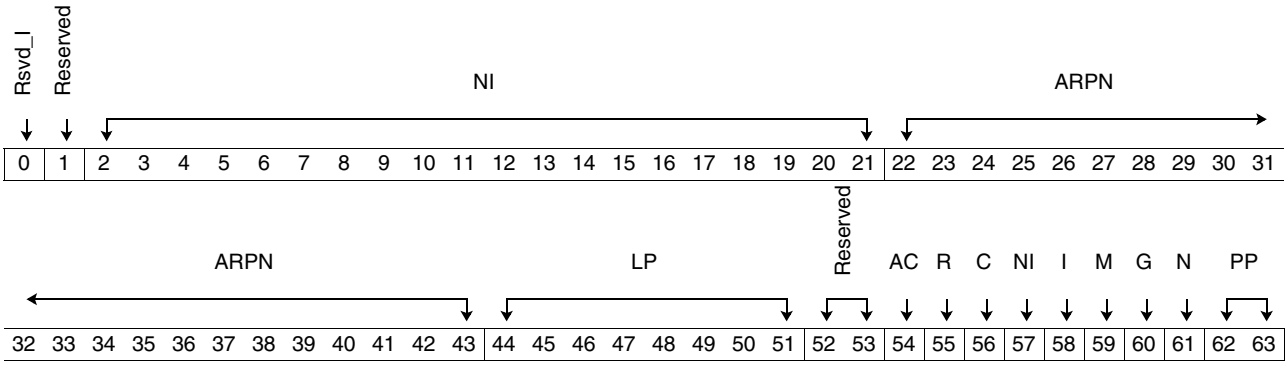
3.2.3.5 MFC TLB Real Page Number Register (MFC_TLB_RPN)

The MFC_TLB_RPN register contains the real page number used in translating the effective address to a real address.

To write a new entry in the TLB via MMIO, the MFC_TLB_Index register should be written first. The index points to the target congruence class and entry. The RPN data should be written after the index, followed by a write to the VPN register. RPN and VPN write data is collected, and then the entire TLB entry is written at once. Any write to the VPN register sets off a write to the TLB array, whether the Index and RPN were previously written or not.

To read the MFC_TLB_RPN data, first write the MFC_TLB_Index to specify the entry. The RPN read command retrieves data from the TLB array entry specified by the MFC_TLB_Index value.

Register Short Name	MFC_TLB_RPN	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400518’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	x‘00000000_00000110’	**Value During POR Set By**	Scan initialization during POR Bits 55, 59 hardwired
Specification Type	CBEA architected register	**Unit**	SMM



Bit(s)	Field Name	Description
0	Rsvd_I	Reserved. Latch bit is implemented; value read is the value written.
1	Reserved	Bits are not implemented; all bits read back zero.
2:21	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
22:43	ARPN	Abbreviated real page number

Bit(s)	Field Name	Description
44:51	LP	Size selector for a large virtual page. This field supports up to eight concurrent large page sizes. The MFC supports three large page sizes in total, but only two concurrently. The two concurrent sizes are encoded in the SMM_HID register. This LP field selects one of those two page sizes for the intended TLB entry. The least signficant bit of this field is used as part of the comparison to the virtual address on a TLB lookup. Depending on the page size, some bits of this field may be concatenated with the ARPN field to form the RPN on a successful lookup. aaaaaaaa MFC_TLB_VPN[L] = '0' aaaaaaa0 Large page size one if MFC_TLB_VPN[L] = '1' aaaaaa01 Large page size two if MFC_TLB_VPN[L] = '1'
52:53	Reserved	Bits are not implemented; all bits read back zero.
54	AC	Address compare bit
55	R	Reference bit. This bit is not written into the TLB array because it is assumed to be '1'. It is returned as '1' during an MMIO RPN read.
56	C	Change bit
57	NI	This bit is defined in the *Cell Broadband Engine Architecture* and is not implemented in the CBE.
58	I	Cache-inhibited page bit
59	M	Memory-coherency storage-control bit This bit is not written into the TLB array because it is assumed to be '1'. It is returned as '1' during an MMIO RPN read. The SBI sends local bus command requests with M='1'. However, the EIB can detect whether it is a local (on-chip) address by comparison to the EIB_LBAR0 and EIB_LBAR1 registers and then forcing M='0'. See the EIB Registers section of this document for more information.
60	G	Guarded page bit
61	N	No-execute page bit
62:63	PP	Page protection bits

3.2.3.6 MFC TLB Invalidate Entry Register (MFC_TLB_Invalidate_Entry)

The MFC_TLB_Invalidate_Entry register is used to invalidate TLB entries in the MFC. The function of this register is similar to the PowerPC **tlbie** instruction. Access to this register is privileged.

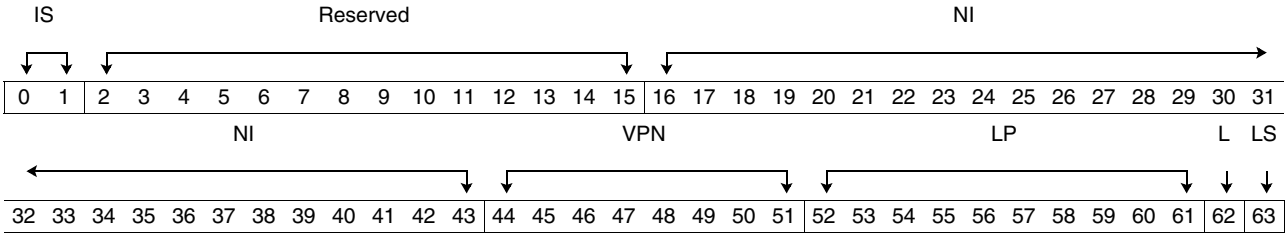
The MFC_TLB_Invalidate_Entry register contains a virtual-page number (VPN) field and an invalidation selector (IS) field. The VPN is used to identify a particular entry to invalidate, and the IS field is used to control how selective the invalidate should be.

This register is not available for the PPE. To invalidate entries in a PPE, the local form of the PowerPC **tlbie** instruction should be used. See the *PowerPC Architecture* document for details of this instruction.

Notes:

- If the VPN is being invalidated to change the protection attributes of a page or to steal the page, a TLB invalidate entry command must be issued to invalidate any cache of the effective-to-real address translation that may be associated with the TLB entry being invalidated. The IS field in the MFC TLB invalidate entry register can only be used to invalidate the cache and does not affect TLB entries.
- Care must be taken in using this function in TLB-managed environments, since hardware may invalidate all TLB entries in the associated congruence class. This could adversely affect TLB set management and deterministic response. To avoid this side effect, privileged software can use the TLB index and TLB direct-modification functions to locate the specific entry to be invalidated in the congruence class and only invalidate the entry that matches.

Register Short Name	MFC_TLB_Invalidate_Entry	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400540’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	N/A	**Value During POR Set By**	N/A
Specification Type	CBEA architected register	**Unit**	SMM



Bit(s)	Field Name	Description
0:1	IS	Invalidation selector 01 The TLB entry is not invalidated. Any lower-level caches of the translation are invalidated. 00, 10, 11 The TLB does a congruency-class invalidate, regardless of logical partition identification (LPID) match.
2:15	Reserved	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
16:43	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
44:51	VPN	Virtual page number (Index)

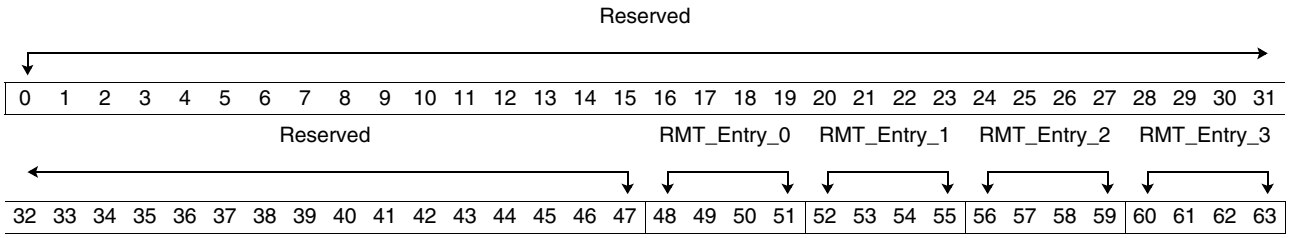
Bit(s)	Field Name	Description
52:61	LP	Size selector for a large virtual page. This field supports up to eight concurrent large page sizes. The MFC supports three large page sizes in total, but only two concurrently. The two concurrent sizes are encoded in the SMM_HID register. This LP field selects one of those two pages. Depending on the page size selected, some bits in this field can be concatenated with the VPN field to determine which entries are invalidated. aaaaaaaa TLB_Invalidate_Entry[L] = '0' aaaaaaa0 Large page size one if TLB_Invalidate_Entry[L] = '1' aaaaaa01 Large page size two if TLB_Invalidate_Entry[L] = '1' **Note:** Software should set the least significant bit of the LP field to the same value as the LS bit.
62	L	Large Page bit The L and LS bits of MFC_TLB_Invalidate_Entry are used in conjunction with the Page Size Decode bits in the SMM_HID register to determine the large page size. 0 Small page (4 KB) 1 Large pages (64 KB, 1 MB, or 16 MB)
63	LS	Large Page Selection The L and LS bits of MFC_TLB_Invalidate_Entry are used in conjunction with the Page Size Decode bits in the SMM_HID register to determine the large page size. 0 First large page (implementation-dependent size) 1 Second large page (implementation-dependent size) **Note:** Software should set this bit to the same value as the least significant bit of the LP field for compatibility with implementations that only support two large page sizes.

3.2.4 MFC TLB Replacement Management Table Data Register (MFC_TLB_RMT_Data)

The MFC supports four Replacement Management Table (RMT) entries. The 2 lower bits of the RClassID from a DMA command opcode select one of the four RMT table entries. The RMT entries use 4 bits to determine which TLB entry to replace within a congruence class during a tablewalk. See the *Cache Replacement Management Facility* section of the *Cell Broadband Engine Architecture* document for more information.

The SMM implements Replacement Management Table (RMT) functionality in the same manner as the PPE.

Register Short Name	MFC_TLB_RMT_Data	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400710’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM



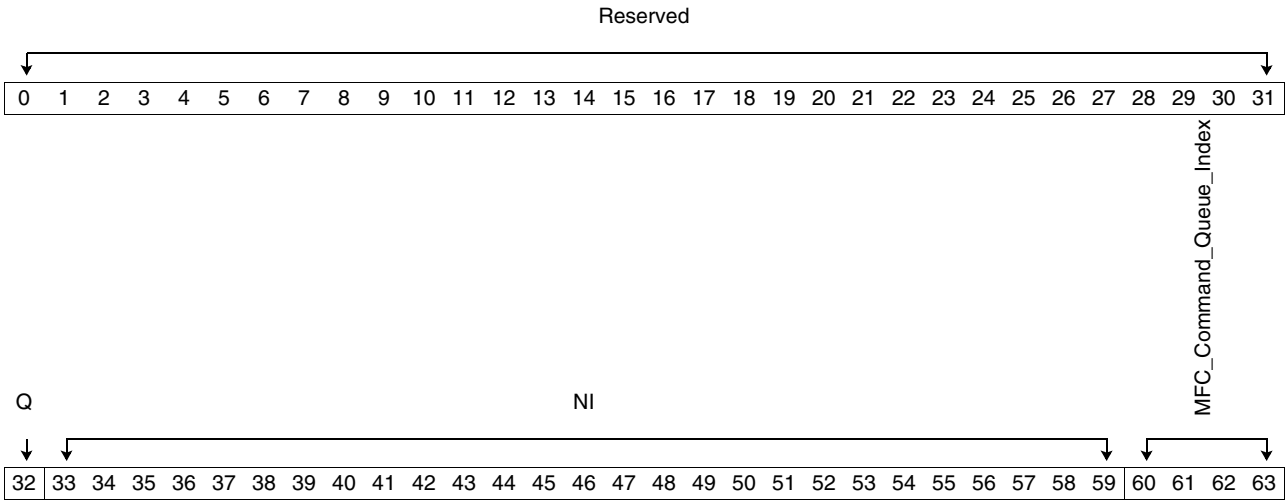
Bit(s)	Field Name	Description
0:47	Reserved	Bits are not implemented; all bits read back zero.
48:51	RMT_Entry_0	RMT Entry 0 set enable bits
52:55	RMT_Entry_1	RMT Entry 1 set enable bits
56:59	RMT_Entry_2	RMT Entry 2 set enable bits
60:63	RMT_Entry_3	RMT Entry 3 set enable bits

3.2.5 MFC Command Data-Storage Interrupt (DSI) Registers

3.2.5.1 MFC Data-Storage Interrupt Pointer Register (MFC_DSIPR)

The MFC_DSIPR Register contains the index (pointer) to the command in the selected command queue that has an MFC data-storage interrupt (DSI) or an MFC data-segment interrupt. The cause of an MFC data-storage interrupt is identified in the *MFC Data-Storage Interrupt Pointer Register (MFC_DSIPR)*.

Register Short Name	MFC_DSIPR	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400800’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32	Q	MFC Command Queue Index 0 MFC Proxy Command Queue index 1 MFC SPU Command Queue index
33:59	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
60:63	MFC_Command_Queue_Index	Points to the queue entry that caused the data-storage interrupt. The number of bits implemented in this field is implementation dependent: 8 bits for the MFC Proxy Command Queue, 16 bits for the MFC SPU Command Queue. Bit [60] is ‘0’ for the MFC SPU Command Queue ID.

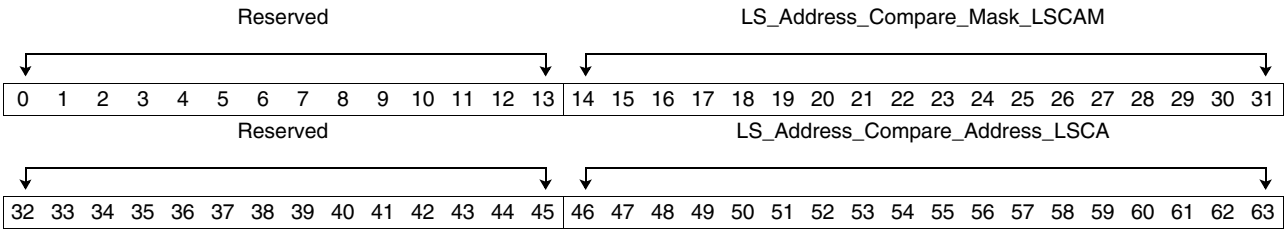
Implementation Note: Only one translation fault may be outstanding. The implementation can either stop all command queue processing on the first translation error or continue processing. If processing is continued, all ordering rules must be followed (a command must not be processed if it is dependent on a command that is waiting for a translation fault to be resolved). The state of the MFC must appear as if the command (or partial command) were never issued. This is also the case if a second translation fault occurs.

3.2.5.2 MFC Local Store Address Compare Register (MFC_LSACR)

The MFC Local Store Address Compare Register (MFC_LSACR) contains the local store address and local store address mask to be used in the MFC local store address compare operation selected by the MFC_ACCR register. Access to the MFC_LSACR register is privileged.

A local store address compare occurs when the local store address accessed is within the range of addresses specified by the bit-wise AND of the local store compare address mask (LSCAM) and the local storage compare address (LSCA) fields.

Register Short Name	MFC_LSACR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400808’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SBI



Bit(s)	Field Name	Description
0:13	Reserved	Bits are not implemented; all bits read back zero.
14:31	LS_Address_Compare_ Mask_LSCAM	The local store address compare mask used in the LS address compare operation
32:45	Reserved	Bits are not implemented; all bits read back zero.
46:63	LS_Address_Compare_ Address_LSCA	The local store compare address used in the LS address compare operation

3.2.5.3 MFC Local Store Compare Results Register (MFC_LSCRR)

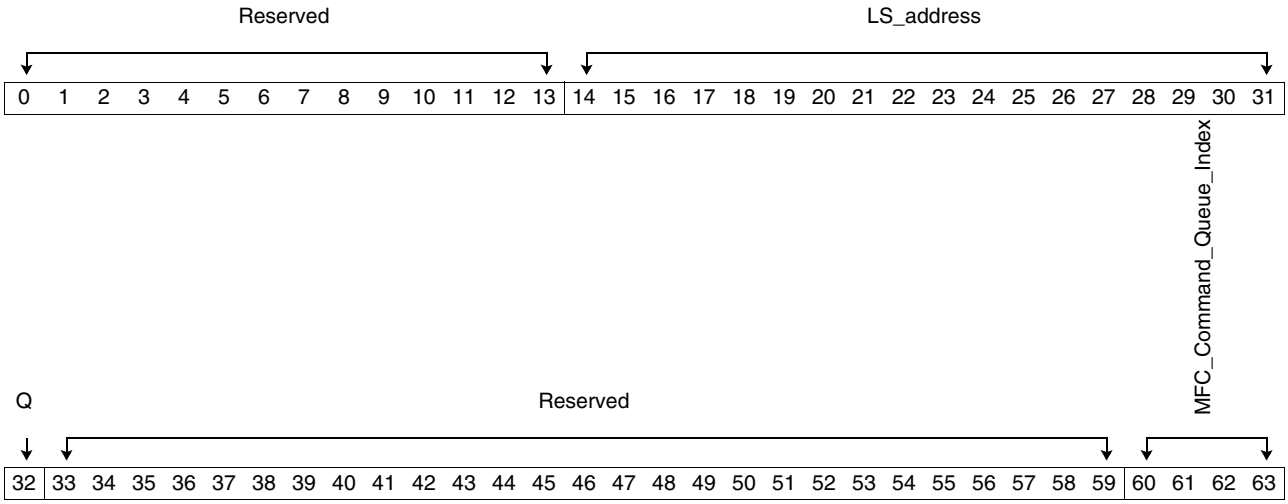
The MFC_LSCRR register contains the local store address that triggered the compare, as well as the MFC command queue index of the DMA command that triggered the compare stop.

Contents of this register are only valid when a class 1 interrupt occurs with the INT_Mask_class1[LP] bit or INT_Mask_class1[LG] bit, or both interrupt status bits set. The MFC_LSCRR[Q] bit indicates whether the command was issued from the MFC proxy command queue or the MFC SPU command queue.

Access to this register should be privileged. The contents of this register become indeterminate once MFC operation is resumed.

A new value is captured and locked into the register by the first qualified compare match. The register is unlocked and ready to capture the next value after the register is read or a DMA purge is issued to the MFC_CNTL[PC] register.

Register Short Name	MFC_LSCRR	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400810’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SBI



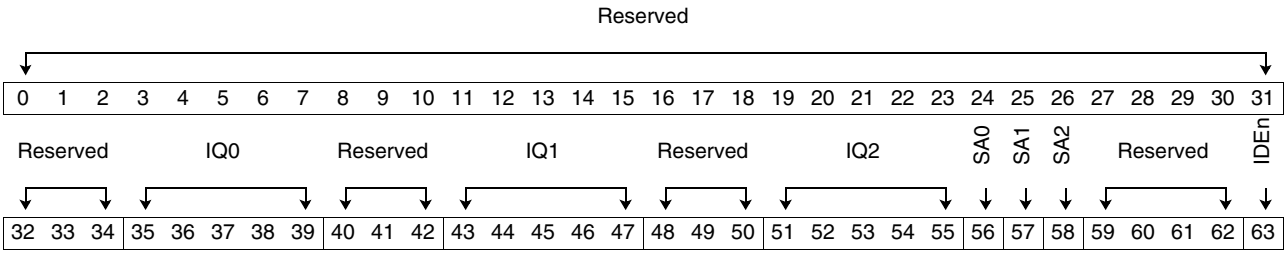
Bit(s)	Field Name	Description
0:13	Reserved	Bits are not implemented; all bits read back zero.
14:31	LS_address	The local store address that triggered the compare match
32	Q	MFC command type 0 Compare match is triggered by an MFC Proxy command. 1 Compare match is triggered by an MFC SPU command.
33:59	Reserved	Bits are not implemented; all bits read back zero.
60:63	MFC_Command_Queue_Index	Points to the MFC Command Queue entry that triggered the compare match. Bit [60] is always zero for the MFC Proxy Command Queue index ID.

3.2.5.4 MFC Transfer Class ID Register (MFC_TClassID)

When slot alternation is enabled, the **put** command is placed into slot 0 and the **get** command is placed into slot 1. When slot alternation is disabled, the corresponding Transfer Class ID group is always placed in and issued from slot 0.

The outgoing queue size for bus transactions is 16, so the total of the values in IQ0, IQ1, and IQ2 must be clamped to 16.

Register Short Name	MFC_TClassID	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400820’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	x‘00000000_00001000’	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC



Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32:34	Reserved	Bits are not implemented; all bits read back zero.
35:39	IQ0	Issue quota for TClassID0. Initial value after purge is ‘10000’. 00000 quota value of 1 00001 quota value of 1 00010 - 10000 quota value of 2 - 16 10001 - 11111 quota value of 16
40:42	Reserved	Bits are not implemented; all bits read back zero.
43:47	IQ1	Issue quota for TClassID1 00000 quota value of 1 00001 quota value of 1 00010 - 10000 quota value of 2 - 16 10001 - 11111 quota value of 16
48:50	Reserved	Bits are not implemented; all bits read back zero.
51:55	IQ2	Issue quota for TClassID2 00000 quota value of 1 00001 quota value of 1 00010 - 10000 quota value of 2 - 16 10001 - 11111 quota value of 16
56	SA0	Slot alternation 0 Enabled for TClassID0 1 Disabled for TClassID0

Bit(s)	Field Name	Description
57	SA1	Slot alternation 0 Enabled for TClassID1 1 Disabled for TClassID1
58	SA2	Slot alternation 0 Enabled for TClassID2 1 Disabled for TClassID2
59:62	Reserved	Bits are not implemented; all bits read back zero.
63	IDEn	TClassID enable bit. Also called the streaming hint enable bit 0 Transfer class ID ignored; all transfers default to TClassID0. 1 Bus transactions issued round-robin for TClassID0, TClassID1, and TClassID2.

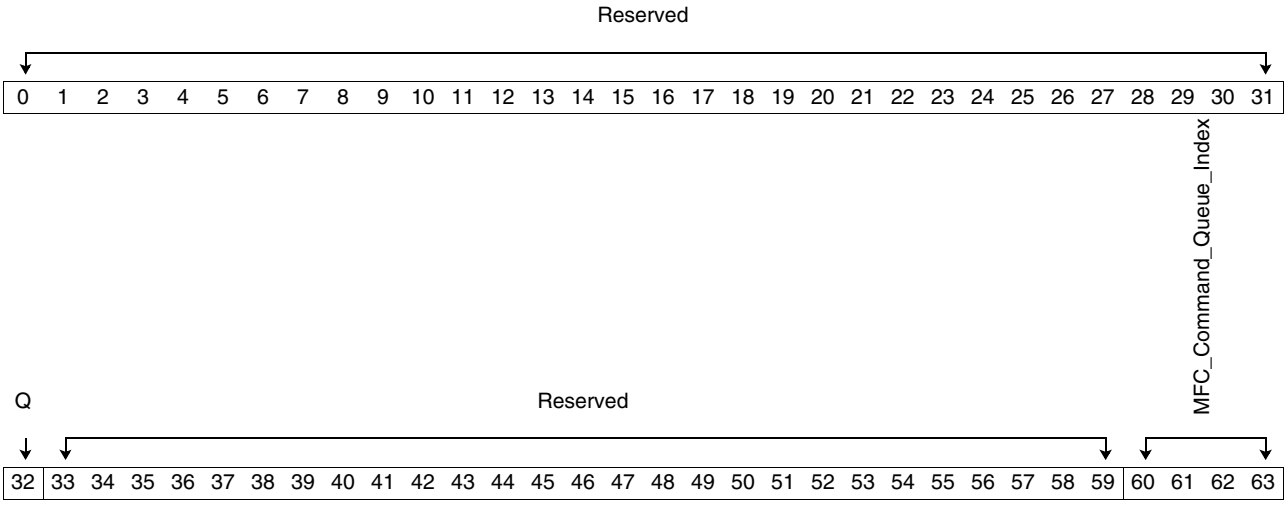
3.2.6 MFC Command Error Area

3.2.6.1 MFC Command Error Register (MFC_CER)

The MFC_CER register contains the MFC command queue entry index of the command that generated the invalid DMA command interrupt or the DMA alignment interrupt.

The MFC must stop execution on the first error. The MFC_CER register must point to the command that caused the first error.

Register Short Name	MFC_CER	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400C00’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	SBI



Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32	Q	MFC Command Queue index 0 MFC Proxy Command Queue index 1 MFC SPU Command Queue index
33:59	Reserved	Bits are not implemented; all bits read back zero.
60:63	MFC_Command_Queue_Index	Points to the MFC Command Queue entry that caused the command error. Bit [60] is always ‘0’ for the MFC Proxy Command Queue index.

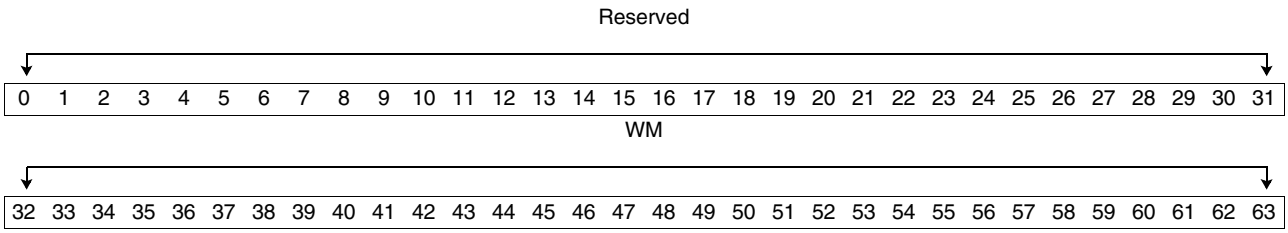
3.2.7 MFC Performance Monitor Register

3.2.7.1 Performance Monitor/Trace Tag Status Wait Mask Register (PM_Trace_Tag_Wait_Mask)

The wait mask is applied to the tag status information in the MFC_RdTagStat channel. If all tag status bits are set to 0 (the tag group has outstanding operations or has been disabled by the query mask), then the mask waits for bits to be set to '1' (the tag group has no outstanding operations or was not disabled by the query mask).

In this wait mask register, bit [32] corresponds to tag group 31, and bit [63] corresponds to tag group 0.

Register Short Name	PM_Trace_Tag_Wait_Mask	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x'401400' + (x'02000' x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC



Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32:63	WM	Wait mask (for each bit in field) 0 Waiting on all masked tags. 1 At least one masked tag is complete.

3.3 SPE Privilege 2 Memory Map Registers

This section lists the registers included in the SPE Privilege 2 memory map.

3.3.1 SLB Management Registers

These registers maintain the SLB array contents.

3.3.1.1 SLB Index Register (SLB_Index)

To properly load the array with data, the SLB requires a write sequence similar to the one for the TLB. First, the index must be written to specify the entry for loading. The VSID and ESID fields are written independently, unlike the TLB writes. The SLB_VSID write should follow the index. The SLB_ESID data is written last because it contains the Valid bit, and the entry should not be valid until all data is loaded.

The CBE implements the three least signficant bits for the MMU. These registers are not available for the PPE.

See the *Cell Broadband Engine Architecture* for more information about this register.

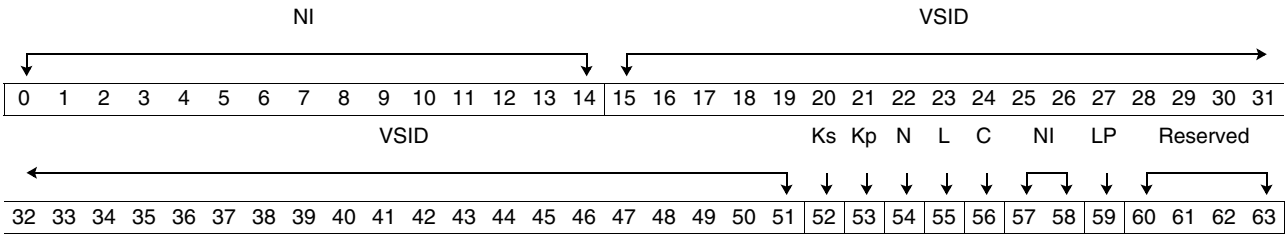
Register Short Name	SLB_Index	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘061108’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM

Bit(s)	Field Name	Description
0:51	Reserved	Bits are not implemented; all bits read back zero.
52:60	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
61:63	SLB_Index	Bits are encoded to select one of eight SLB entries. The SLB_Index-implemented bits are defined in the range [61:63] (three bits total). The eight possible Index values are the eight binary combinations of those three bits: ‘000’, ‘001’, ‘010’, ‘011’ ... The SLB array entries are most commonly referred to by the decimal equivalent of those binary numbers: 0, 1, 2, ..., 7.

3.3.1.2 SLB Virtual Segment ID Register (SLB_VSID)

To properly load the array with data, the SLB requires a write sequence similar to the one for the TLB. First, the index must be written to specify the entry for loading. The VSID and ESID fields are written independently, unlike the TLB writes. The SLB_VSID write should follow the index. The SLB_ESID data is written last because it contains the Valid bit, and the entry should not be valid until all data is loaded.

Register Short Name	SLB_VSID	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘061118’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM

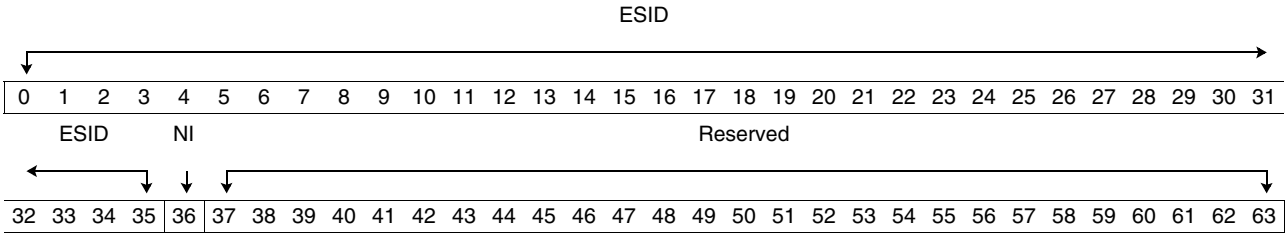


Bit(s)	Field Name	Description
0:14	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
15:51	VSID	Virtual Segment ID
52	Ks	Storage Key supervisor (privileged) state
53	Kp	Storage Key problem state
54	N	No Execute Segment
55	L	Large page bit 1 Large page (64 KB, 1 MB, 16 MB) 0 Small page (4 KB)
56	C	Class bit
57:58	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
59	LP	Large page-size bit
60:63	Reserved	Bits are not implemented; all bits read back zero.

3.3.1.3 SLB Invalidate Entry Register (SLB_Invalidate_Entry)

Software uses this register to maintain the SLB and update entries.

Register Short Name	SLB_Invalidate_Entry	**Privilege Type**	Privilege 2
Access Type	MMIO Write Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘061120’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	SMM



Bit(s)	Field Name	Description
0:35	ESID	Effective Segment ID
36	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
37:63	Reserved	Bits are not implemented; all bits read back zero.

3.3.2 Context Save and Restore Register

The hardware supports the capability to suspend a task on the SPE, fully save its context, fully restore that context at a later time, and resume the task.

3.3.2.1 MFC Command Queue Context Save/Restore Register (MFC_CQ_SR)

Table 3-4 through *Table 3-10* define the MFC Command Queue Context Save/Restore Register.

Register Short Name	MFC_CQ_SR	**Privilege Type**	Privileged
Access Type	MMIO Write Only	**Width**	64 bits
Hex Offset Address	x‘02000’ -‘022FF’ See *Table 3-4* through *Table 3-10*	**From Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	SMF

Context for the MFC command queue and issue logic is accessed starting at offset x‘2000’, as shown in *Table 3-4* on page 57. The MFC command queues contain 16 entries for the SPU and eight for the PPU, and each MFC command queue entry is accessed with three MMIO doublewords. The contents of doubleword 0 (DW0) are listed in *Table 3-5* on page 58. The contents of doubleword 1 (DW1) are listed in *Table 3-6* on page 59. The contents of doubleword 2 (DW2) are listed in *Table 3-7* on page 59.

The rightmost two columns of *Table 3-4* describe the fourth doubleword for each entry in the MFC command queue. For the first 16 entries, the doubleword is divided into the SPU Issue Word (SIW) and the Tag Completion Word. For the last eight entries, the doubleword is divided into the PPU Issue Word (PIW) and a Reserved Word. The format of the SPU Issue Word is described in *Table 3-8*. The format of the PPU Issue Word is described in *Table 3-9*. *Table 3-10* describes the data in the Tag Completion Word for the first 16 entries. The Tag Completion Word contains the counts for two tag groups for the SPU and PPU, the stall, list, and finish bits for each SPU entry, and the start and finish bits for each PPU entry (only in the first eight Tag Completion Words).

See the *Cell Broadband Engine Architecture* for more general information about Context Save and Restore.

Table 3-4. Definitions for the MFC Context Save/Restore Registers (Read/Write) (Page 1 of 2)

Address offset	+ x'000(CMDQ_DW0)[1]	+ x'008(CMDQ_DW1)[2]	+ x'010u(CMDQ_DW2)[3]	+ x'018 (Issue_Word) (Issue Machine States) SIW[4]/PIW[5]	+ x'01c (Tag_word)
x'02000'	MFC SPUQ Entry #0, DW0	MFC SPUQ Entry #0, DW1	MFC SPUQ Entry #0, DW2	MFC SPUQ Entry #0, SIW	See *Section 3-10* on page 60
x'02020'	MFC SPUQ Entry #1, DW0	MFC SPUQ Entry #1, DW1	MFC SPUQ Entry #1, DW2	MFC SPUQ Entry #1, SIW	See *Section 3-10* on page 60
x'02040'	MFC SPUQ Entry #2, DW0	MFC SPUQ Entry #2, DW1	MFC SPUQ Entry #2, DW2	MFC SPUQ Entry #2, SIW	See *Section 3-10* on page 60
x'02060'	MFC SPUQ Entry #3, DW0	MFC SPUQ Entry #3, DW1	MFC SPUQ Entry #3, DW2	MFC SPUQ Entry #3, SIW	See *Section 3-10* on page 60
x'02080'	MFC SPUQ Entry #4, DW0	MFC SPUQ Entry #4, DW1	MFC SPUQ Entry #4, DW2	MFC SPUQ Entry #4, SIW	See *Section 3-10* on page 60
x'020a0'	MFC SPUQ Entry #5, DW0	MFC SPUQ Entry #5, DW1	MFC SPUQ Entry #5, DW2	MFC SPUQ Entry #5, SIW	See *Section 3-10* on page 60
x'020c0'	MFC SPUQ Entry #6, DW0	MFC SPUQ Entry #6, DW1	MFC SPUQ Entry #6, DW2	MFC SPUQ Entry #6, SIW	See *Section 3-10* on page 60
x'020e0'	MFC SPUQ Entry #7, DW0	MFC SPUQ Entry #7, DW1	MFC SPUQ Entry #7, DW2	MFC SPUQ Entry #7, SIW	See *Section 3-10* on page 60
x'02100'	MFC SPUQ Entry #8, DW0	MFC SPUQ Entry #8, DW1	MFC SPUQ Entry #8, DW2	MFC SPUQ Entry #8, SIW	See *Section 3-10* on page 60
x'02120'	MFC SPUQ Entry #9, DW0	MFC SPUQ Entry #9, DW1	MFC SPUQ Entry #9, DW2	MFC SPUQ Entry #9, SIW	See *Section 3-10* on page 60
x'02140'	MFC SPUQ Entry #a, DW0	MFC SPUQ Entry #a, DW1	MFC SPUQ Entry #a, DW2	MFC SPUQ Entry #a, SIW	See *Section 3-10* on page 60
x'02160'	MFC SPUQ Entry #b, DW0	MFC SPUQ Entry #b, DW1	MFC SPUQ Entry #b, DW2	MFC SPUQ Entry #b, SIW	See *Section 3-10* on page 60
x'02180'	MFC SPUQ Entry #c, DW0	MFC SPUQ Entry #c, DW1	MFC SPUQ Entry #c, DW2	MFC SPUQ Entry #c, SIW	See *Section 3-10* on page 60
x'021a0'	MFC SPUQ Entry #d, DW0	MFC SPUQ Entry #d, DW1	MFC SPUQ Entry #d, DW2	MFC SPUQ Entry #d, SIW	See *Section 3-10* on page 60
x'021c0'	MFC SPUQ Entry #e, DW0	MFC SPUQ Entry #e, DW1	MFC SPUQ Entry #e, DW2	MFC SPUQ Entry #e, SIW	See *Section 3-10* on page 60

1. For a definition of this format, see *Table 3-5 Context Save/Restore MFC Command Queue Doubleword 0 (CSR_CMDQ_DW0)* on page 58.
2. For a definition of this format, see *Table 3-6 Context Save/Restore MFC Command Queue Doubleword 1 (CSR_CMDQ_DW1)* on page 59.
3. For a definition of this format, see *Table 3-7 Context Save/Restore MFC Command Queue Doubleword 2 (CSR_CMDQ_DW02)* on page 59.
4. For a definition of this format, see *Table 3-8 Context Save/Restore for MFC SPU Command Queue Issue State Machine* on page 59.
5. For a definition of this format, see *Table 3-9 Context Save/Restore for MFC Proxy Command Queue Issue Machine State* on page 59.

Table 3-4. Definitions for the MFC Context Save/Restore Registers (Read/Write) (Page 2 of 2)

Address offset	+ x'000(CMDQ_DW0)[1]	+ x'008(CMDQ_DW1)[2]	+ x'010u(CMDQ_DW2)[3]	+ x'018 (Issue_Word) (Issue Machine States) SIW[4]/PIW[5]	+ x'01c (Tag_word)
x'021e0'	MFC SPUQ Entry #f, DW0	MFC SPUQ Entry #f, DW1	MFC SPUQ Entry #f, DW2	MFC SPUQ Entry #f, SIW	See *Section 3-10* on page 60
x'02200'	MFC PrxyQ Entry #0, DW0	MFC PrxyQ Entry #0, DW1	MFC PrxyQ Entry #0, DW2	MFC PrxyQ Entry #0, PIW	Reserved
x'02220'	MFC PrxyQ Entry #1, DW0	MFC PrxyQ Entry #1, DW1	MFC PrxyQ Entry #1, DW2	MFC PrxyQ Entry #1, PIW	Reserved
x'02240'	MFC PrxyQ Entry #2, DW0	MFC PrxyQ Entry #2, DW1	MFC PrxyQ Entry #2, DW2	MFC PrxyQ Entry #2, PIW	Reserved
x'02260'	MFC PrxyQ Entry #3, DW0	MFC PrxyQ Entry #3, DW1	MFC PrxyQ Entry #3, DW2	MFC PrxyQ Entry #3, PIW	Reserved
x'02280'	MFC PrxyQ Entry #4, DW0	MFC PrxyQ Entry #4, DW1	MFC PrxyQ Entry #4, DW2	MFC PrxyQ Entry #4, PIW	Reserved
x'022a0'	MFC PrxyQ Entry #5, DW0	MFC PrxyQ Entry #5, DW1	MFC PrxyQ Entry #5, DW2	MFC PrxyQ Entry #5, PIW	Reserved
x'022c0'	MFC PrxyQ Entry #6, DW0	MFC PrxyQ Entry #6, DW1	MFC PrxyQ Entry #6, DW2	MFC PrxyQ Entry #6, PIW	Reserved
x'022e0'	MFC PrxyQ Entry #7, DW0	MFC PrxyQ Entry #7, DW1	MFC PrxyQ Entry #7, DW2	MFC PrxyQ Entry #7, PIW	Reserved

1. For a definition of this format, see *Table 3-5 Context Save/Restore MFC Command Queue Doubleword 0 (CSR_CMDQ_DW0)* on page 58.
2. For a definition of this format, see *Table 3-6 Context Save/Restore MFC Command Queue Doubleword 1 (CSR_CMDQ_DW1)* on page 59.
3. For a definition of this format, see *Table 3-7 Context Save/Restore MFC Command Queue Doubleword 2 (CSR_CMDQ_DW02)* on page 59.
4. For a definition of this format, see *Table 3-8 Context Save/Restore for MFC SPU Command Queue Issue State Machine* on page 59.
5. For a definition of this format, see *Table 3-9 Context Save/Restore for MFC Proxy Command Queue Issue Machine State* on page 59.

Table 3-5. Context Save/Restore MFC Command Queue Doubleword 0 (CSR_CMDQ_DW0)

CSR_CMDQ_DW0	Description
0:14	List Address[0:14]
15:26	List Size[0:11]
27:34	MFC Command Opcode[0:7]
35:39	MFC Command Tag[0:4]
40	List Valid Bit
41:43	RclassID[0:2]
44:46	TclassID[0:2]
47:63	Reserved

Table 3-6. Context Save/Restore MFC Command Queue Doubleword 1 (CSR_CMDQ_DW1)

CS_CMDQ_DW1	Description
0:51	Effective Address (EA)[0:51]
52:63	Reserved

Table 3-7. Context Save/Restore MFC Command Queue Doubleword 2 (CSR_CMDQ_DW02)

CS_CMDQ_DW2	Description
0:13	Local-Store Address[0:13]
14:24	Transfer Size[0:10]
25:36	Effective Address[52:63]
37	No-op Valid Bit
38	Quadword (QW) or Multiple QW Valid Bit
39	EA Valid Bit
40	Command Error Bit
41:63	Reserved

Table 3-8. Context Save/Restore for MFC SPU Command Queue Issue State Machine

CS_SPU_Issue_Word	Description
0:15	Entry Dependency State
16	Entry Valid
17:18	TclassID
19:23	MFC Command Tag[0:4]
24	0 DMA **get** command 1 DMA **put** command
25	Last Bit
26:27	Dependency Type 00 Normal 01 MFC command with barrier modifier 10 Barrier, **mfcsync**, or **mfceieio** DMA command 11 Reserved
28	Stall Bit
29	Issue Dependency Bit
30:31	Reserved

Table 3-9. Context Save/Restore for MFC Proxy Command Queue Issue Machine State

CS_Prxy_Issue_Word	Description
0:7	Entry Dependency State
8:15	Reserved
16	Entry Valid
17:18	TclassID
19:23	MFC Command Tag[0:4]

Table 3-9. Context Save/Restore for MFC Proxy Command Queue Issue Machine State

CS_Prxy_Issue_Word	Description
24	0 DMA **get** command 1 DMA **put** command
25	Last Bit
26:27	Dependency Type 00 Normal 01 MFC command with barrier modifier 10 Barrier, **mfcsync**, or **mfceieio** MFC command 11 Reserved
28	Reserved
29	Issue Dependency Bit
30:31	Reserved

Table 3-10. Context Save/Restore for Tag Completion Machine State

Word Address	SPU Tag Group Count		PPU Tag Group Count		MFC SPUQ Stall Bit	MFC SPUQ List Bit	MFC SPUQ Finish Bit	MFC ProxyQ Start Bit	MFC ProxyQ Finish Bit	Not Used
Bit position in Word	[0:4]	[5:9]	[10:13]	[14:17]	[18]	[19]	[20]	[21]	[22]	[23:31]
x‘0201C’	tag 0	tag 1	tag 0	tag 1	Entry #0	Entry #0	Entry #0	Entry #0	Entry #0	Reserved
x‘0203C’	tag 2	tag 3	tag 2	tag 3	Entry #1	Entry #1	Entry #1	Entry #1	Entry #1	Reserved
x‘0205C’	tag 4	tag 5	tag 4	tag 5	Entry #2	Entry #2	Entry #2	Entry #2	Entry #2	Reserved
x‘0207C’	tag 6	tag 7	tag 6	tag 7	Entry #3	Entry #3	Entry #3	Entry #3	Entry #3	Reserved
x‘0209C’	tag 8	tag 9	tag 8	tag 9	Entry #4	Entry #4	Entry #4	Entry #4	Entry #4	Reserved
x‘020BC’	tag 10	tag 11	tag 10	tag 11	Entry #5	Entry #5	Entry #5	Entry #5	Entry #5	Reserved
x‘020DC’	tag 12	tag 13	tag 12	tag 13	Entry #6	Entry #6	Entry #6	Entry #6	Entry #6	Reserved
x‘020FC’	tag 14	tag 15	tag 14	tag 15	Entry #7	Entry #7	Entry #7	Entry #7	Entry #7	Reserved
x‘0211C’	tag 16	tag 17	tag 16	tag 17	Entry #8	Entry #8	Entry #8	Reserved	Reserved	Reserved
x‘0213C’	tag 18	tag 19	tag 18	tag 19	Entry #9	Entry #9	Entry #9	Reserved	Reserved	Reserved
x‘0215C’	tag 20	tag 21	tag 20	tag 21	Entry #10	Entry #10	Entry #10	Reserved	Reserved	Reserved
x‘0217C’	tag 22	tag 23	tag 22	tag 23	Entry #11	Entry #11	Entry #11	Reserved	Reserved	Reserved
x‘0219C’	tag 24	tag 25	tag 24	tag 25	Entry #12	Entry #12	Entry #12	Reserved	Reserved	Reserved
x‘021BC’	tag 26	tag 27	tag 26	tag 27	Entry #13	Entry #13	Entry #13	Reserved	Reserved	Reserved
x‘021DC’	tag 28	tag 29	tag 28	tag 29	Entry #14	Entry #14	Entry #14	Reserved	Reserved	Reserved
x‘021FC’	tag 30	tag 31	tag 30	tag 31	Entry #15	Entry #15	Entry #15	Reserved	Reserved	Reserved

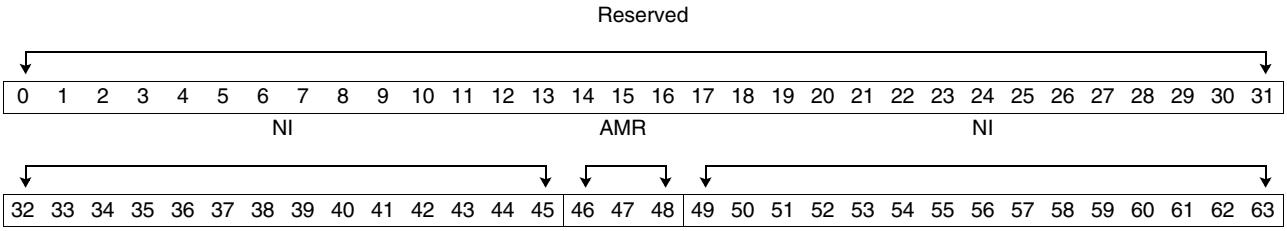
See the *Cell Broadband Engine Architecture* for more information about this register.

3.3.2.2 SPU Local Store Limit Register (SPU_LSLR)

Access to this register is privileged. The SPU_LSLR register provides privileged software with a means to artificially limit the size of local store available to an application. This provides for backwards compatibility with applications sensitive to the size of local store. The value written to this register limits the size of the local store. If an application performs a quadword load or store from the SPU beyond the range of the SPU_LSLR, the operation occurs at the resulting wrapped address. The default value initialized at POR is a local-store address limit of 256 KB. In SPU isolate mode, the address mask register (AMR) value, SPU_LSLR[46:48], is forced to '111', and writes have no effect. This register can only be updated while the SPU is stopped, as indicated in SPU_Status[31].

Note: Reads and writes can be to the full 64 bits or to only the lower-word bits [32:63].

Register Short Name	SPU_LSLR	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x'064058' + (x'80000' x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	x'00000000_0003FFFF'	**Value During POR Set By**	Scan initialization during POR Bits 49-63 hardwired
Specification Type	CBEA architected register	**Unit**	MFC

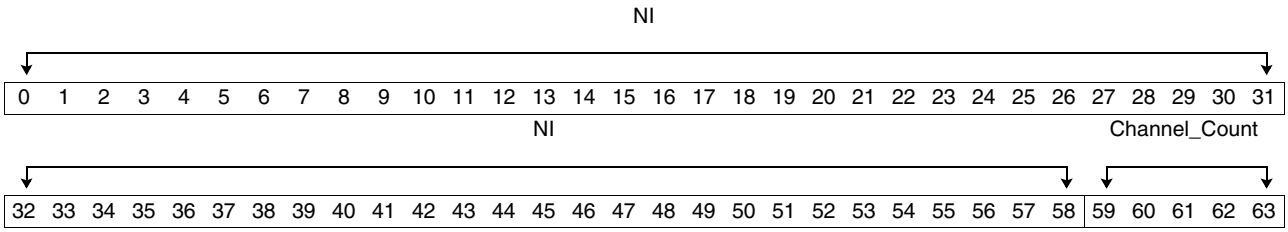


Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32:45	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
46:48	AMR	Address Mask Register (AMR) Local Store Limit Range 111 256-KB Local Store access limit 011 128-KB Local Store access limit 001 64-KB Local Store access limit 000 32-KB Local Store access limit Other combinations are invalid.
49:63	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. Reads always return ones.

3.3.2.3 SPU Channel Count Register (SPU_ChnlCnt)

The SPU_ChnlCnt register is used to read or initialize the count associated with the channel selected by the *SPU Channel Index Register (SPU_ChnlIndex)*.

Register Short Name	SPU_ChnlCnt	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘064068’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	N/A	**Value During POR Set By**	N/A
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:31	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
32:58	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
59:63	Channel_Count	The 5-bit channel count used by software. Usually set to 0, 1, or 16. See the **Programming Note**.

Programming Note: It is recommended that channel counts be initialized by privileged software before a new context is started in an SPU, as shown in the following table:

Channel Names	Recommended Initialization for the Channel Count Setting
SPU_RdEventStat channel SPU_RdSigNotify1 channel SPU_RdSigNotify2 channel MFC_RdTagStat channel MFC_RdListStallStat channel MFC_RdAtomicStat channel SPU_RdInMbox channel	Initialize to 0
MFC_WrMSSyncReq channel MFC_WrTagUpdate channel MFC_WrOutMbox channel MFC_WrOutIntrMbox channel	Initialize to 1
MFC_Cmd channel	Initialize to 16

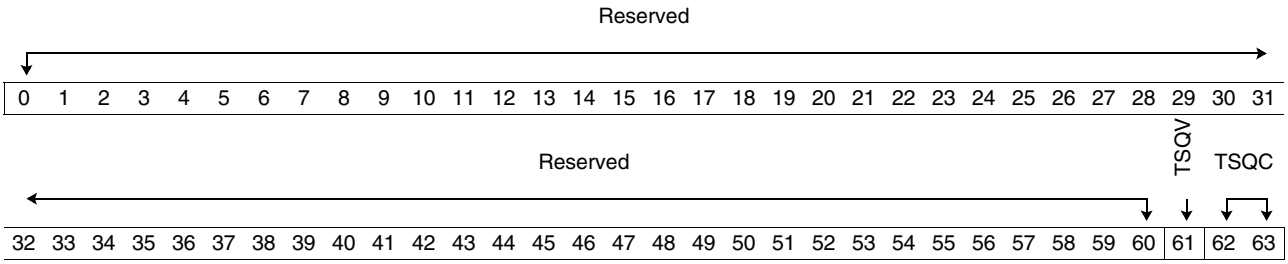
3.3.3 Context Save and Restore Registers (Implementation-Specific)

There are four implementation-specific context save and restore registers.

3.3.3.1 Context Save and Restore for SPU MFC Commands Register (MFC_CSR_TSQ)

This register stores context data for the state of the Tag Status Query logic in the DMAC unit of the MFC.

Register Short Name	MFC_CSR_TSQ	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘065008’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC

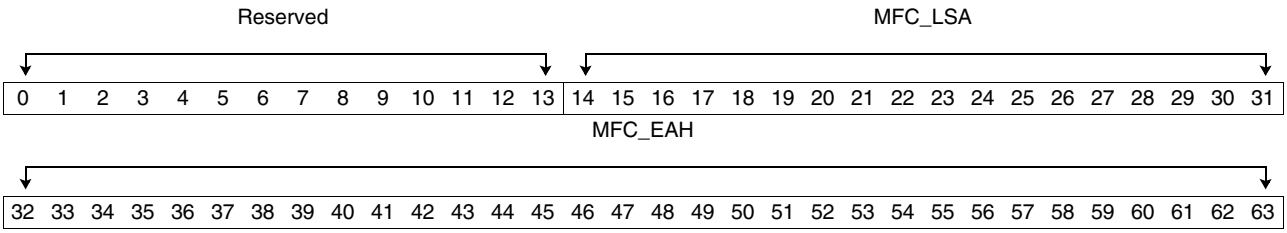


Bit(s)	Field Name	Description
0:60	Reserved	Bits are not implemented; all bits read back zero.
61	TSQV	Tag Status Query Valid TSQV = 1 & TSQC = x‘01’ Waiting for ANY condition to be met TSQV = 1 & TSQC = x‘10’ Waiting for ALL conditions to be met TSQV = 0 No pending query **Note:** A valid query with Immediate Condition ‘00’ should never be saved as context since the query is always completed prior to a context save.
62:63	TSQC	Tag Status Query Condition

3.3.3.2 Context Save and Restore for SPU MFC Commands Register (MFC_CSR_CMD1)

MFC_CSR_CMD1 and MFC_CSR_CMD2 are used for context save and restore operations. MFC_CSR_CMD1 stores the data for the MFC_LSA channel and the MFC_EAH channel.

Register Short Name	MFC_CSR_CMD1	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘065010’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC

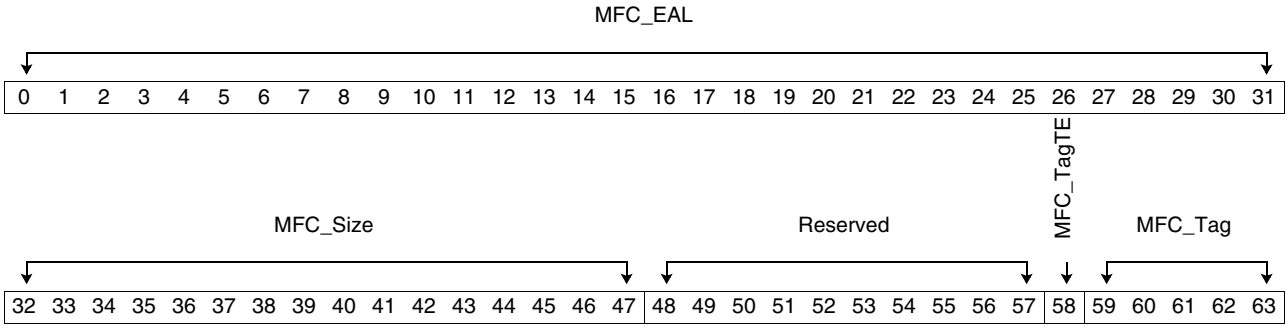


Bit(s)	Field Name	Description
0:13	Reserved	Bits are not implemented; all bits read back zero.
14:31	MFC_LSA	MFC Local Store Address
32:63	MFC_EAH	MFC Effective Address High

3.3.3.3 Context Save and Restore for SPU MFC Commands Register (MFC_CSR_CMD2)

MFC_CSR_CMD1 and MFC_CSR_CMD2 are used for context save and restore operations. MFC_CSR_CMD2 stores the data for the MFC_EAL channel, the MFC_Size channel, and the MFC_TagID channel.

Register Short Name	MFC_CSR_CMD2	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘065018’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC



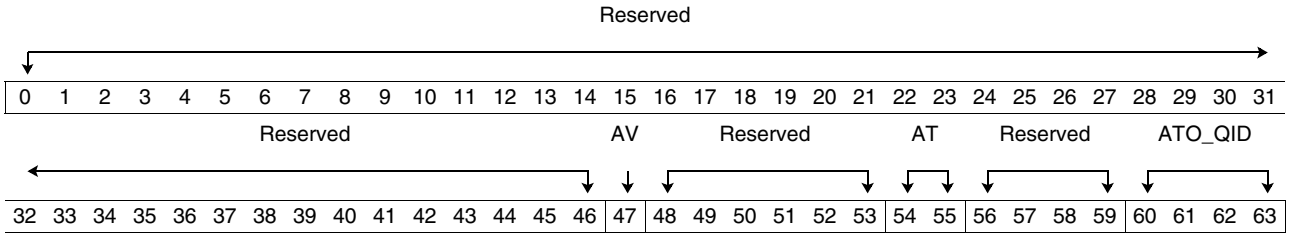
Bit(s)	Field Name	Description
0:31	MFC_EAL	MFC Effective Address Low
32:47	MFC_Size	MFC Transfer Size
48:57	Reserved	Bits are not implemented; all bits read back zero.
58	MFC_TagTE	MFC Tag data [0:26] is nonzero. A tag error causes an MFC command error interrupt.
59:63	MFC_Tag	MFC Tag ID data [27:31]

3.3.3.4 Context Save and Restore for SPU Atomic Immediate Command (MFC_CSR_ATO)

This register stores the state of the Atomic Immediate command in the MFC SPUQ.

The Atomic Immediate command is only valid for context saves when the command has not yet gone to the Atomic Unit for processing. Any Atomic command that has gone to the Atomic unit is completed prior to context save and is invalidated.

Register Short Name	MFC_CSR_ATO	**Privilege Type**	Privilege 2
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘065020’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Privilege 2
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC



Bit(s)	Field Name	Description
0:46	Reserved	Bits are not implemented; all bits read back zero.
47	AV	Atomic Valid
48:53	Reserved	Bits are not implemented; all bits read back zero.
54:55	AT	Atomic Type 00 No atomic immediate command 01 **getllar** 10 **putllc** 11 **putlluc**
56:59	Reserved	Bits are not implemented; all bits read back zero.
60:63	ATO_QID	Entry ID of the MFC SPU Command Queue holding the Atomic command

3.4 SPE Problem State Memory Map Registers

This section lists the registers included in the Problem State memory map.

3.4.1 MFC Multisource Synchronization Register

3.4.1.1 MFC Multisource Synchronization Register (MFC_MSSync)

This register is the interface to the MFC Multisource Synchronization facility. See the *Cell Broadband Engine Architecture* for more information on this facility. Writing any value to this register requests a synchronization. At the time of the write, the MFC starts to track all outstanding transfers targeting the corresponding SPE. When read, this register returns the current status of the last request. A value of zero is returned when all transfers targeting the SPE and received before the last write of the MFC_MSSync register are complete.

Register Short Name	MFC_MSSync	**Privilege Type**	Problem State
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘040000’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MFC

Bit(s)	Field Name	Description
0:62	Reserved	Bits are not implemented; all bits read back zero.
63	P	Pending 0 All transfers prior to writing the MFC_MSSync register are complete. 1 All transfers prior to writing the MFC_MSSync register are not complete.

3.4.2 MFC Command Parameter Registers

These registers describe the MFC Command Parameter channels.

3.4.2.1 MFC Local Store Address Register (MFC_LSA)

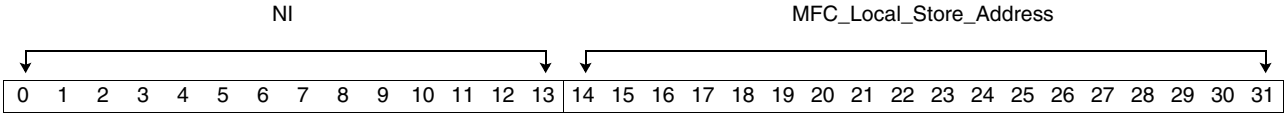
The MFC local store address parameter stored in the MFC_LSA register is used to supply the SPU local store address associated with a DMA command to be queued. This address is used as the source or destination of the DMA transfer as it is defined in the DMA command.

The contents of the MFC local store address parameter are not persistent and must be written for each DMA command-enqueued sequence.

The validity of this parameter is checked asynchronous to the instruction stream. If the address is unaligned, MFC command queue processing is suspended, and an MFC DMA alignment exception is generated.

Note: Providing a local store address above the implemented range of local store causes the local store address to wrap around to a valid address, but no exception indicates that this condition has occurred.

Register Short Name	MFC_LSA	**Privilege Type**	Problem State
Access Type	MMIO Write Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘043004’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:13	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
14:31	MFC_Local_Store_Address	MFC_LSA[25:31] must always be aligned to a transfer-size boundary and the 4 least-significant bits of the local-store address must match the 4 least-significant bits of the effective address.

3.4.2.2 MFC Effective Address High Register (MFC_EAH)

The validity of this parameter is checked asynchronous to the instruction stream. If a segment fault, mapping fault, or protection violation occurs, an MFC data segment exception is generated. If the address is not aligned, an MFC DMA alignment exception is generated.

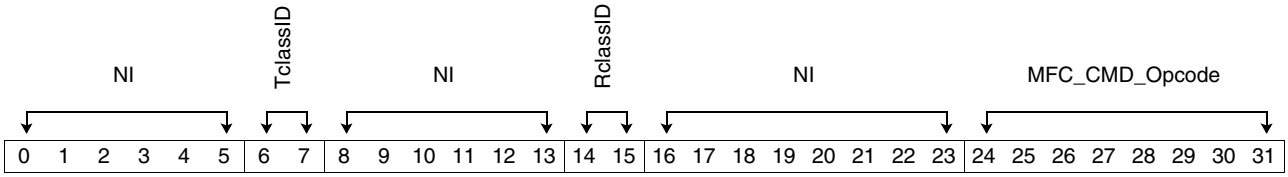
See the *Cell Broadband Engine Architecture* for more information about this register.

3.4.2.3 MFC Class ID Command Opcode Register (MFC_ClassID_CMD)

This register is documented in the *Cell Broadband Engine Architecture* as two registers, MFC_ClassID and MFC_CMD. The MFC class ID parameter is used to specify the replacement class ID and the transfer class ID for each MFC command. The transfer class ID (TclassID) bit field is used to identify access to storage with differing characteristics.

The MFC_ClassID_CMD register, which is write only, is related to the MFC_CMDStatus register, which is read only. See *MFC Command Status Register (MFC_CMDStatus)* on page 71 for more information.

Register Short Name	MFC_ClassID_CMD	**Privilege Type**	Problem State
Access Type	MMIO Write Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘043014’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:5	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
6:7	TclassID	Transfer class identifier
8:13	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
14:15	RclassID	Replacement class identifier
16:23	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
24:31	MFC_CMD_Opcode	MFC command opcode

Programming Note: The total number of queue slots is implementation-specific and varies between implementations. For portability of an application, the enqueue sequence for MFC commands and the method to determine the number of queue slots available should be provided as a macro.

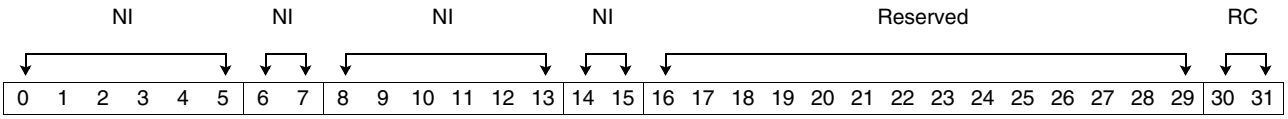
3.4.2.4 MFC Command Status Register (MFC_CMDStatus)

The MFC_CMDStatus register contains the return code from the last attempt to enqueue an MFC command. The return code is read from the same location as the command, tag, and size.

Note: The MFC_CMDStatus register is a read-only, 32-bit register; the most-significant 16 bits are implementation specific. The MFC command return code in the least-significant bits returns the command status when read.

The MFC_CMDStatus register, which is read only, is related to the MFC_ClassID_CMD register, which is write only. See *MFC Class ID Command Opcode Register (MFC_ClassID_CMD)* on page 70 for more information.

Register Short Name	MFC_CMDStatus	**Privilege Type**	Problem State
Access Type	MMIO Read Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘043014’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:5	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
6:7	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. Bits return the last value written to this address (the TClass ID field, bits [6:7] of the MFC_ClassID_CMD register).
8:13	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
14:15	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. Bits return the last value written to this address (the RClass ID field, bits [14:15] of the MFC_ClassID_CMD register).
16:29	Reserved	Bits are not implemented; all bits read back zero.
30:31	RC	MFC command return code 00 Command enqueue successful. 01 Command enqueue failed due to sequencing error. 10 Command enqueue failed due to insufficient space in the command queue (the free space in the command queue is zero). 11 Command enqueue failed due to sequencing error, and free space in the command queue is zero.

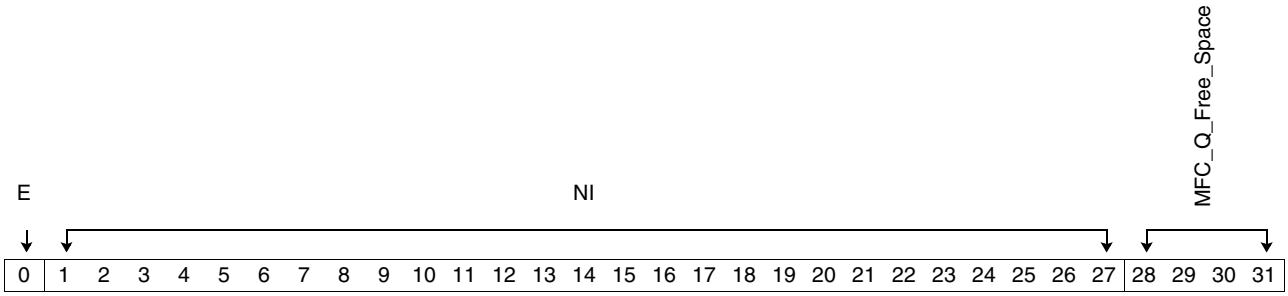
3.4.3 MFC Proxy Command Queue Control Registers

The registers in this section are used to control the MFC proxy command queue.

3.4.3.1 MFC Queue Status Register (MFC_QStatus)

The MFC_QStatus register contains the current status of the MFC command queue. MFC_QStatus[0] indicates whether the MFC proxy command queue is empty or contains valid commands that are not yet complete. The least-significant 4 bits of this register return the number of entries available in the MFC proxy command queue. A value of zero in this field indicates that the queue is full.

Register Short Name	MFC_QStatus	**Privilege Type**	Problem State
Access Type	MMIO Read Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘043104’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC

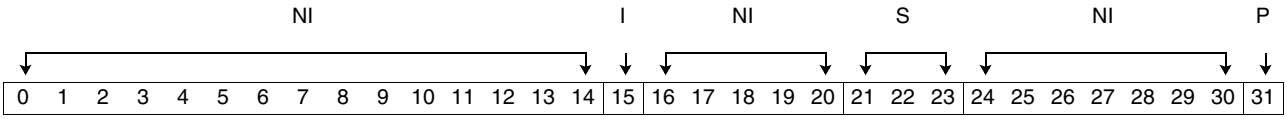


Bit(s)	Field Name	Description
0	E	MFC proxy command queue empty. All MFC operations are complete. 0 MFC Proxy Command queue contains commands 1 MFC Proxy Command queue does not contain commands
1:27	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
28:31	MFC_Q_Free_Space	MFC queue free space This field contains the number of queue entries available. Software can use this field to set a loop count for the number of MFC commands to enqueue. Software must not assume a command is enqueued based on the free space. Other conditions may cause the command issue sequence to fail. See the *Cell Broadband Engine Architecture* for more information.

3.4.3.2 SPU Mailbox Status Register (SPU_Mbox_Stat)

The SPU_Mbox_Stat register contains the current state of the mailbox queues between the SPU and other processors and devices.

Register Short Name	SPU_Mbox_Stat	**Privilege Type**	Problem State
Access Type	MMIO Read Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘044014’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	x‘00000400’	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC

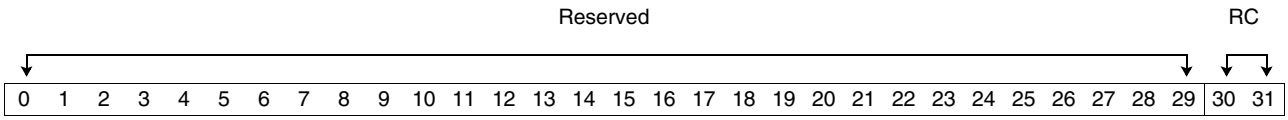


Bit(s)	Field Name	Description
0:14	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
15	I	SPU Outbound Interrupt Mailbox status is equal to 1 minus the SPU_WrOutIntrMbox channel count. This status bit is set when the SPU Outbound Interrupt Mailbox is written by the SPU. It is reset when the PPU reads the SPU Outbound Interrupt Mailbox 0 SPU Outbound Interrupt Mailbox is empty. 1 SPU Outbound Interrupt Mailbox contains a new value.
16:20	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
21:23	S	SPU Inbound Mailbox status is equal to 4 minus the SPU_RdInMbox channel count. This status bit decrements when the SPU Inbound Mailbox is written by the PPU. It increments when the SPU reads the SPU Inbound Mailbox. 000 SPU Inbound Mailbox is full. 001 SPU Inbound Mailbox has one location available to load. 010 SPU Inbound Mailbox has two locations available to load. 011 SPU Inbound Mailbox has three locations available to load. 100 SPU Inbound Mailbox has four locations available to load. ‘101’ through ‘111’ are unused.
24:30	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
31	P	SPU Outbound Mailbox status is equal to 1 minus the SPU_WrOutMbox channel count. This status bit is set when the SPU Outbound Mailbox is written by the SPU. It is reset when the PPU reads the SPU Outbound Mailbox. 0 SPU Outbound Mailbox is empty. 1 SPU Outbound Mailbox contains a new value.

3.4.3.3 SPU Run Control Register (SPU_RunCntl)

The SPU_RunCntl Register is used to start and stop the execution of instructions in the SPU. The SPU can dynamically change the state of the run bit. The current status of the SPU run state is available in the *SPU Status Register (SPU_Status)*. When this register is read, it returns the last data written for the last valid write.

Register Short Name	SPU_RunCntl	**Privilege Type**	Problem State
Access Type	MMIO Read/Write	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘04401C’ + (x‘80000’ x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:29	Reserved	Bits are not implemented; all bits read back zero.
30:31	RC	SPU run control 00 SPU stop request. No instructions are issued. 01 SPU run request. Instruction is issued if not stalled on condition. 10 SPU isolate exit request. 11 SPU isolate load request. The current status of the SPU run state is available in the SPU_Status register.

Programming Note: After SPU_RunCntl[30:31], the run control bit, is set to stop the SPU, the SPU is not stopped until SPU_Status[31], the run status bit, reads ‘0’. See the *SPU Status Register (SPU_Status)* for more information. A write of ‘01’ while the SPU is idle causes the SPU to restart from the Program Counter at which it stopped (including in Isolation state), except if the SPU stopped from an error condition (illegal opcode or uncorrectable ECC) while in Isolation state. Then, if the SPU is stopped in isolation state because of an error, a write of ‘01’ has no effect. A write of ‘10’ or ‘11’ is required to restart the SPU. The SPU ignores writes of ‘01’, ‘10’, or ‘11’ unless the SPU is idle, as indicated when SPU_Status[31], the R bit, reads ‘0’.

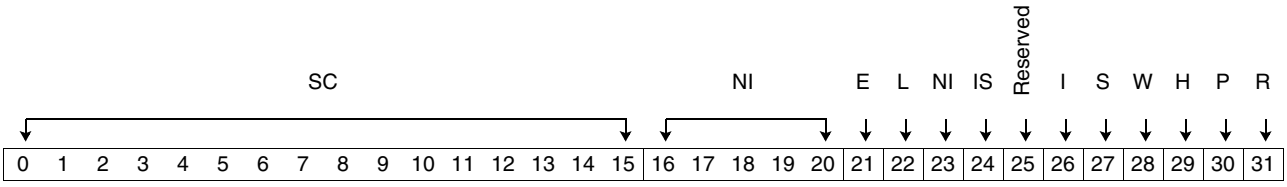
3.4.3.4 SPU Status Register (SPU_Status)

Reading this register provides a snapshot of the current SPU state. The status read can be dynamically changing if the SPU is currently running. If the SPU is stopped or halted, the status remains static until the SPU state is changed by software. If the SPU was stopped under PPU control at the same time that the SPU was waiting on a blocked channel, the SPU Wait status is set in conjunction with the SPU Stopped status. Multiple state bits ([26:27] and [29:30]) may be set based on the program design.

When SPU_Status[31] transitions to '1', the states for SPU_Status[26:27] and SPU_Status[29:30] are reset to '0'.

Note: For more information on the bits below that reference the isolate load or isolate exit states, see the *SPU Isolation Facility* section in the *Cell Broadband Engine Architecture*.

Register Short Name	SPU_Status	**Privilege Type**	Problem State
Access Type	MMIO Read Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x'044024' + (x'80000' x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:15	SC	If SPU_Status[30], the P bit, which is the stop and signal indication, is set to '1', this field provides a copy of bits [18-31] of the SPU stop and signal instruction that caused the SPU stop. Bits [0:1] of this field are always set to '0'. If SPU_Status[30], the P bit, is not set, data in this field is not valid. A stop and signal with dependencies (**stopd**) instruction, used for debugging, always sets each of bits [2:15] to '1'.
16:20	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
21	E	SPU isolate exit status 0 The SPU is not performing an isolate exit. 1 The SPU is performing an isolate exit.
22	L	SPU isolate load status 0 The SPU is not performing an isolate load. 1 The SPU is performing an isolate load.
23	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
24	IS	SPU isolation status 0 The SPU is in a nonisolated state. 1 The SPU is in an isolated state.
25	Reserved	Bit is not implemented; bit reads back zero.

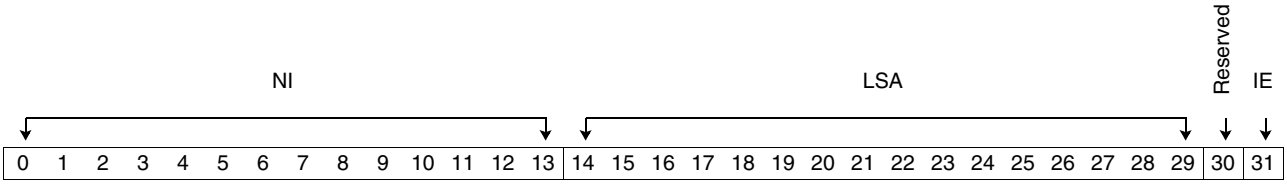
Bit(s)	Field Name	Description
26	I	Invalid Instruction Detected. The SPU does not stop precisely, and the *SPU Next Program Counter Register (SPU_NPC)* may not indicate the instruction after the illegal instruction. 0 No illegal opcodes have been issued. 1 An illegal opcode has been issued, and the SPU has been halted. An SPU error interrupt is also generated in INT_Class0[61] if enabled in SPU_ERR_Mask[63].
27	S	SPU Single-Step Status. The *SPU Next Program Counter Register (SPU_NPC)* points to the next instruction after the instructions committed for the single-step operation. 0 SPU not stopped due to single-step mode. 1 SPU stopped after one completed instruction in single-issue mode or a pair of instructions in dual-issue mode.
28	W	SPU Wait Status 0 SPU is not waiting on a blocked channel. 1 SPU is waiting on a blocked channel.
29	H	SPU Halt Status. The SPU does not stop precisely, and the *SPU Next Program Counter Register (SPU_NPC)* may not indicate the instruction after the halt instruction. SPU_Status[29] is not set if SPU stops due to Single Step. 0 SPU is not halted due to a **halt** instruction. 1 SPU is halted due to a **halt** instruction.
30	P	SPU Program Stop and Signal Status. The *SPU Next Program Counter Register (SPU_NPC)* points to the instruction after the commited stop instruction. 0 SPU is not stopped due to a **Stop and Signal** instruction. 1 SPU is stopped due to a **Stop and Signal** instruction.
31	R	SPU Run Status 0 SPU stopped (idle) 1 SPU running

3.4.3.5 SPU Next Program Counter Register (SPU_NPC)

The Local Store Address (LSA) value read from this register is limited by the value of SPU_LSLR[46:48], the AMR field.

A read from this register is only valid when the SPU is stopped (SPU_Status[31] is set to '0') and the SPU is in non-isolate state. Otherwise, it returns meaningless data (all zeros). Values written to this register while the SPU is running or in Isolate State have no effect on the operation of the SPU and are ignored. SPU Interrupts can only be enabled in this register prior to starting the SPU and cannot be enabled while the SPU is running. That is, the internal enable bit gets its value from this register when the SPU starts running. When the SPU is stopped, the internal enable bit is loaded to this register and may have been changed during program execution by an indirect branch.

Register Short Name	SPU_NPC	**Privilege Type**	Problem State
Access Type	MMIO Read/Write	**Width**	32 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x'044034' + (x'80000' x *n*)	**Memory Map Area**	SPE Problem State
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	CBEA architected register	**Unit**	MFC



Bit(s)	Field Name	Description
0:13	NI	Bits are defined in the *Cell Broadband Engine Architecture*, but are not implemented in the CBE. All bits read back zero.
14:29	LSA	Word-aligned local-store address (LSA).
30	Reserved	Bit is not implemented; bit reads back zero.
31	IE	Interrupt enable state 0 SPU interrupts disabled at start. 1 SPU interrupts enabled at start.

Programming Note: SPU_NPC is not valid for an illegal instruction.

4. BEI I/O Command (IOC) MMIO Registers

This section describes the Cell Broadband Engine Interface (BEI) IOC memory-mapped I/O (MMIO) registers. *Table 4-1* shows the BEI IOC MMIO memory map and lists the BEI IOC registers. The BEI IOC register space starts at x‘511 C00’ and ends at x‘511 FFF’. Offsets are from the start of the BEI IOC register space. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register bit definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

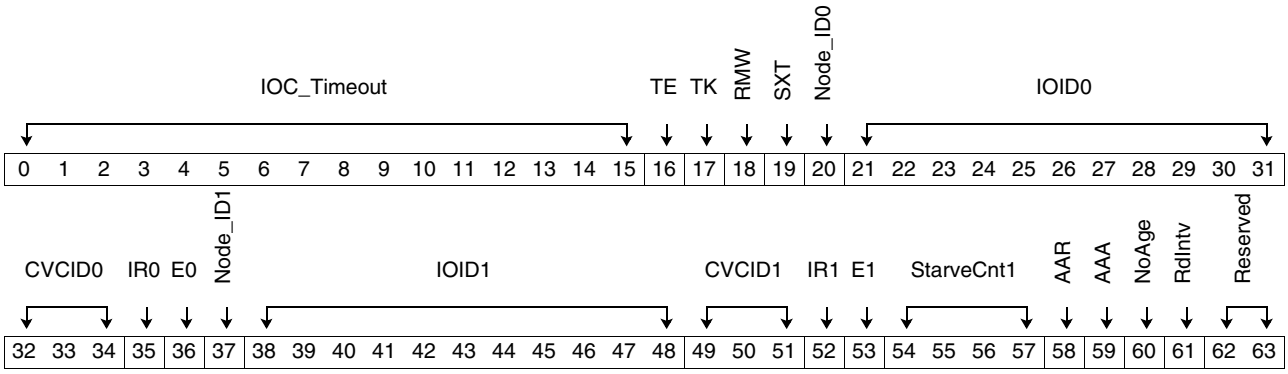
Table 4-1. BEI IOC MMIO Memory Map

Hexadecimal Offset (x‘511 *nnn*’)	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
x‘C00’	*IOC IOCmd Configuration Register (IOC_IOCmd_Cfg)*	64	R/W	*Section 4.1* on page 80
x‘C08’	*IOC Memory Base Address Register (IOC_MemBaseAddr)*	64	R/W	*Section 4.2* on page 82
x‘C10’	*IOC Base Address Register 0 (IOC_BaseAddr0)*	64	R/W	*Section 4.3* on page 83
x‘C18’	*IOC Base Address Mask Register 0 (IOC_BaseAddrMask0)*	64	R/W	*Section 4.4* on page 84
x‘C20’	*IOC Base Address Register 1 (IOC_BaseAddr1)*	64	R/W	*Section 4.5* on page 85
x‘C28’	*IOC Base Address Mask Register 1 (IOC_BaseAddrMask1)*	64	R/W	*Section 4.6* on page 86
x‘C30’ – x‘C50’	Reserved			
x‘C58’	*IOC SRAM Parity Error Capture Register (IOC_SRAM_ParityErrCap)*	64	R	*Section 4.7* on page 87
x‘C60’ – x‘FFF’	Reserved			

4.1 IOC IOCmd Configuration Register (IOC_IOCmd_Cfg)

This register configures basic settings for the IOC.

Register Short Name	IOC_IOCmd_Cfg	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘511C00’	**Memory Map Area**	IOC I/O Command
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	BEI



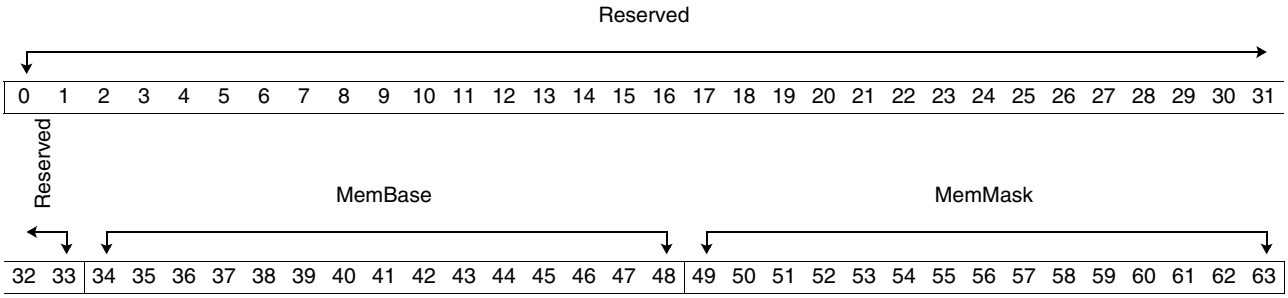
Bits	Field Name	Description
0:15	IOC_Timeout	Timeout time for commands in InCmd or OutCmd. Only the leftmost bit is used. x‘0000’ Disable timeout x‘0001’ 32 µsec x‘0002’ 64 µsec x‘0004’ 128 µsec x‘0008’ 256 µsec x‘0010’ 512 µsec ... x‘8000’ 1 second
16	TE	Enable IO Address Translation
17	TK	Enable Tokens
18	RMW	Read-Modify-Write 0 Get one token for all stores. 1 Get two tokens for stores less than 128 bytes in length.
19	SXT	16 bytes 0 Get one token for all stores. 1 Get two tokens for stores less than 16 bytes in length.
20	Node_ID0	Node ID bit for CBE to use in IOTtags for commands being sent on IOIF0 0 Use node ID 0. 1 Use node ID 1.
21:31	IOID0	I/O ID for CBE to use in commands being sent on I/O interface 0 (IOIF0).
32:34	CVCID0	CVC ID for CBE to use in commands being sent on IOIF0.

Bits	Field Name	Description
35	IR0	Enable broadcast Interrupt Reissue commands seen on the Element Interconnect Bus (EIB) onto IOIF0.
36	E0	Enable broadcast of EIEIO commands seen on the EIB onto IOIF0.
37	Node_ID1	Node ID bit for CBE to use in IOTtags for commands being sent on I/OInterface 1 (IOIF1). 0 Use node ID 0. 1 Use node ID 1.
38:48	IOID1	I/O ID for CBE to use in commands being sent on IOIF1.
49:51	CVCID1	CVC ID for CBE to use in commands being sent on IOIF1.
52	IR1	Enable broadcast Interrupt Reissue commands seen on the EIB onto IOIF1.
53	E1	Enable broadcast of EIEIO commands seen on the EIB onto IOIF1.
54:57	StarveCnt1	Starvation count for IOIF1. IOIF0 has priority over IOIF1 in using the translation logic unless IOIF1 has a command waiting for this field's value of cycles or more.
58	AAR	Token setup: Allow All Requests 0 Requests for IOIFO tokens are not made by a command until it has obtained all its non-IOIF0 tokens. 1 Allow requests for IOIF0 tokens to be made as soon as possible.
59	AAA	Token setup: Allow All Assignments 0 IOIF0 tokens are only assigned to commands that have obtained all their non-IOIF0 tokens. 1 Allow assignments of IOIF0 tokens to be made as soon as possible.
60	NoAge	Token setup: Disable Aging 0 Older commands are favored over newer commands within a Resource Allocation Group (RAG). 1 Disable aging.
61	RdIntv	Read intervention 0 Enable putting a '0' in the N bit on a read of 128 bytes. 1 Enable putting a '1' in the N bit on a read of 128 bytes.
62:63	Reserved	Bits are not implemented; all bits read back zero.

4.2 IOC Memory Base Address Register (IOC_MemBaseAddr)

This register configures the memory base address and mask used by the IOC to determine the real addresses that are mapped to memory. For incoming IOIF reads and writes, the IOC determines whether the associated real address accesses memory. If memory is accessed, the IOC requests the appropriate memory bank token.

Register Short Name	IOC_MemBaseAddr	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘511C08’	**Memory Map Area**	IOC I/O Command
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	BEI

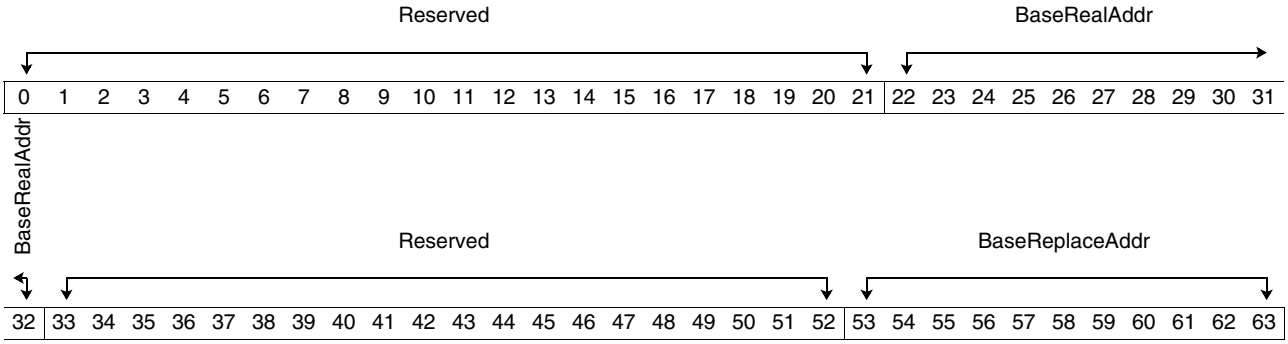


Bits	Field Name	Description
0:33	Reserved	Bits are not implemented; all bits read back zero.
34:48	MemBase	Memory Base Real Address. Assumed to be aligned on a 128-MB boundary. This field represents bits [0:14] of the real memory address.
49:63	MemMask	Memory Mask. Only memory sizes that are powers of two are allowed.

4.3 IOC Base Address Register 0 (IOC_BaseAddr0)

This register configures the address and replacement address for I/O adapter 0.

Register Short Name	IOC_BaseAddr0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘511C10’	**Memory Map Area**	IOC I/O Command
Value at Initial POR	N/A	**Value During POR Set By**	Configuration ring
Specification Type	Implementation-specific register	**Unit**	BEI

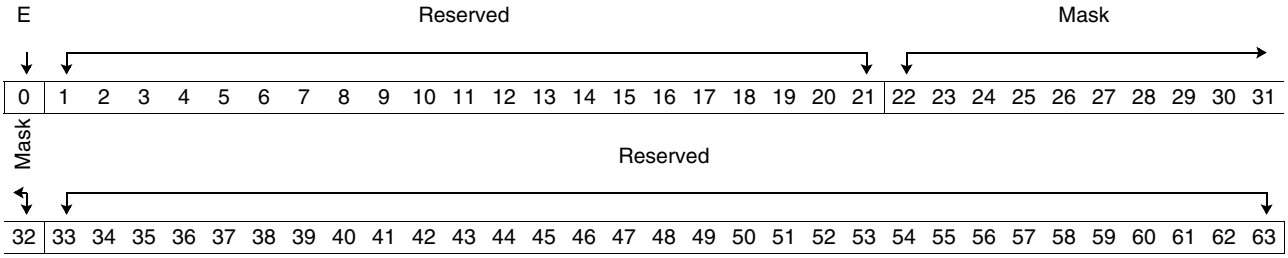


Bits	Field Name	Description
0:21	Reserved	Bits are not implemented; all bits read back zero.
22:32	BaseRealAddr	Base Real Address. Specifies a base real address for accesses that are routed to IOIF0.
33:52	Reserved	Bits are not implemented; all bits read back zero.
53:63	BaseReplaceAddr	Base Replacement Address. Specifies the value to replace part or all of the Base Real Address that is passed to IOIF0.

4.4 IOC Base Address Mask Register 0 (IOC_BaseAddrMask0)

This register configures the address mask for I/O adapter 0.

Register Short Name	IOC_BaseAddrMask0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘511C18’	**Memory Map Area**	IOC I/O Command
Value at Initial POR	N/A	**Value During POR Set By**	Configuration ring
Specification Type	Implementation-specific register	**Unit**	BEI

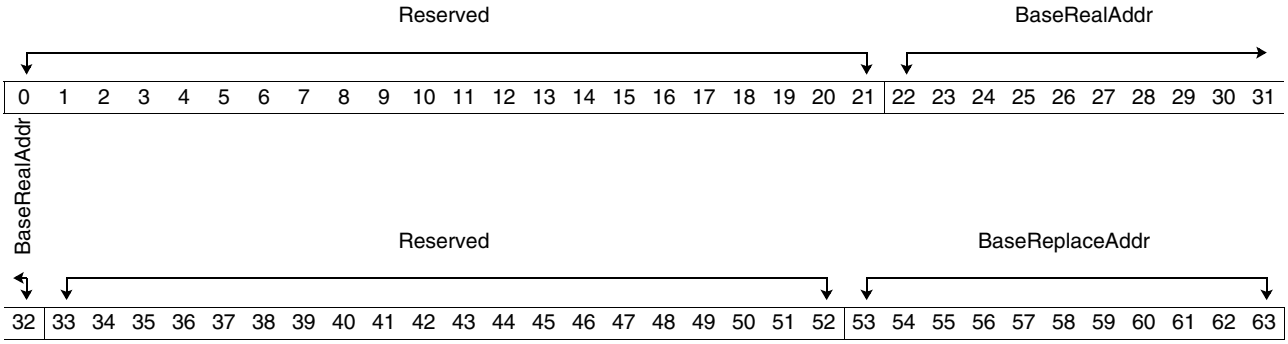


Bits	Field Name	Description
0	E	When set, enables passing accesses to IOIF0.
1:21	Reserved	Bits are not implemented; all bits read back zero.
22:32	Mask	Mask. Specifies the size of the address range that is routed to IOIF0.
33:63	Reserved	Bits are not implemented; all bits read back zero.

4.5 IOC Base Address Register 1 (IOC_BaseAddr1)

This register configures the address and replacement address for I/O adapter 1.

Register Short Name	IOC_BaseAddr1	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511C20'	**Memory Map Area**	IOC I/O Command
Value at Initial POR	N/A	**Value During POR Set By**	Configuration ring
Specification Type	Implementation-specific register	**Unit**	BEI

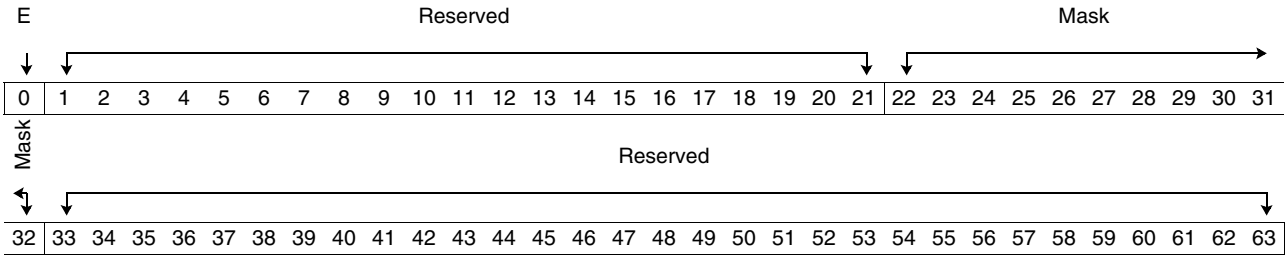


Bits	Field Name	Description
0:21	Reserved	Bits are not implemented; all bits read back zero.
22:32	BaseRealAddr	Base Real Address. Specifies a base real address for accesses that are routed to IOIF1
33:52	Reserved	Bits are not implemented; all bits read back zero.
53:63	BaseReplaceAddr	Base Replacement Address. Specifies the value to replace part or all of the Base Real Address that is passed to IOIF1.

4.6 IOC Base Address Mask Register 1 (IOC_BaseAddrMask1)

This register configures the address mask for I/O adapter 1.

Register Short Name	IOC_BaseAddrMask1	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘511C28’	**Memory Map Area**	IOC I/O Command
Value at Initial POR	N/A	**Value During POR Set By**	Configuration ring
Specification Type	Implementation-specific register	**Unit**	BEI

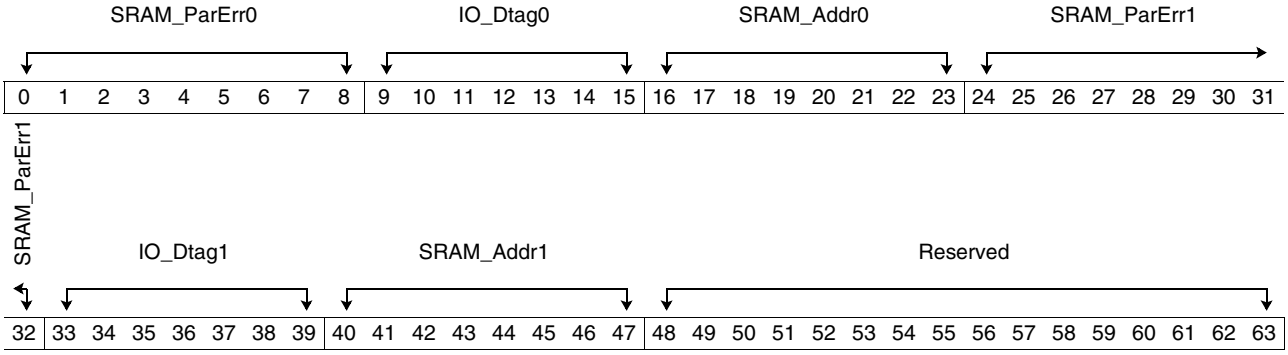


Bits	Field Name	Description
0	E	When set, enables passing accesses to IOIF1.
1:21	Reserved	Bits are not implemented; all bits read back zero.
22:32	Mask	Specifies the size of the address range that is routed to IOIF1.
33:63	Reserved	Bits are not implemented; all bits read back zero.

4.7 IOC SRAM Parity Error Capture Register (IOC_SRAM_ParityErrCap)

This is a read-only register for software that is used to determine the cause of outbound command errors.

Register Short Name	IOC_SRAM_ParityErrCap	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511C58'	**Memory Map Area**	IOC I/O Command
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	BEI



Bits	Field Name	Description
0:8	SRAM_ParErr0	IOIF0 SRAM parity error bits [0:7] HWn (where n = 0-7) parity error bit [8] control field parity error
9:15	IO_Dtag0	IOIF0 I/O tag of command with parity error
16:23	SRAM_Addr0	IOIF0 SRAM address that had parity error
24:32	SRAM_ParErr1	IOIF1 SRAM parity error bits [24:31] HWn (where n = 0-7) parity error bit [32] control field parity error
33:39	IO_Dtag1	IOIF1 I/O tag of command with parity error
40:47	SRAM_Addr1	IOIF1 SRAM address that had parity error
48:63	Reserved	Bits are not implemented; all bits read back zero.

5. IOC Address Translation MMIO Registers

This section describes the I/O Command (IOC) address translation memory-mapped I/O (MMIO) registers. *Table 5-1* shows the IOC address translation MMIO memory map and lists the IOC registers. The IOC address translation space starts at x‘510 000’ and ends at x‘510 FFF’. Offsets are from the start of the IOC address translation register space. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

Table 5-1. IOC Address Translation MMIO Memory Map

Hexadecimal Offset (x‘510 *nnn*’)	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
IOC Address Translation				
x‘000’ – x‘1F8’	*IOC IOPT Cache Directory Register (IOC_IOPT_CacheDir)* (First way)	64	R/W	*Section 5.1* on page 90
x‘200’ – x‘3F8’	*IOC IOPT Cache Directory Register (IOC_IOPT_CacheDir)* (Second way)	64	R/W	*Section 5.1* on page 90
x‘400’ – x‘5F8’	*IOC IOPT Cache Directory Register (IOC_IOPT_CacheDir)* (Third way)	64	R/W	*Section 5.1* on page 90
x‘600’ – x‘7F8’	*IOC IOPT Cache Directory Register (IOC_IOPT_CacheDir)* (Fourth way)	64	R/W	*Section 5.1* on page 90
x‘800’ – x‘8F8’	*IOC IOST Cache Register (IOC_IOST_Cache)*	64	R/W	*Section 5.2* on page 91
x‘900’	*IOC IOST Cache Invalidate Register (IOC_IOST_CacheInvd)*	64	R/W	*Section 5.3* on page 92
x‘908’	*IOC IOPT Cache Invalidate Register (IOC_IOPT_CacheInvd)*	64	R/W	*Section 5.4* on page 93
x‘910’	*IOC IOPT Cache Register (IOC_IOPT_Cache)*	64	R/W	*Section 5.5* on page 94
x‘918’	*IOC I/O Segment Table (IOST) Origin Register (IOC_IOST_Origin)*	64	R/W	*Section 5.6* on page 95
x‘920’	*IOC I/O Exception Status Register (IOC_IO_ExcpStat)*	64	R/W	*Section 5.7* on page 96
x‘928’	*IOC I/O Exception Mask Register (IOC_IO_ExcpMask)*	64	R/W	*Section 5.8* on page 97
x‘930’	*IOC Translation Configuration Register (IOC_XlateCfg)*	64	R/W	*Section 5.9* on page 98
x‘931’ – x‘FFF’	Reserved			

5.1 IOC IOPT Cache Directory Register (IOC_IOPT_CacheDir)

The I/O page table (IOPT) cache and its directory have 64 sets with 4-way associativity, for a total of 256 entries. When an IOPT cache directory entry is read, the IOPT cache register is automatically loaded from the corresponding IOPT cache entry. When an IOPT cache directory entry is written, the corresponding IOPT cache entry is automatically written from the IOPT cache register.

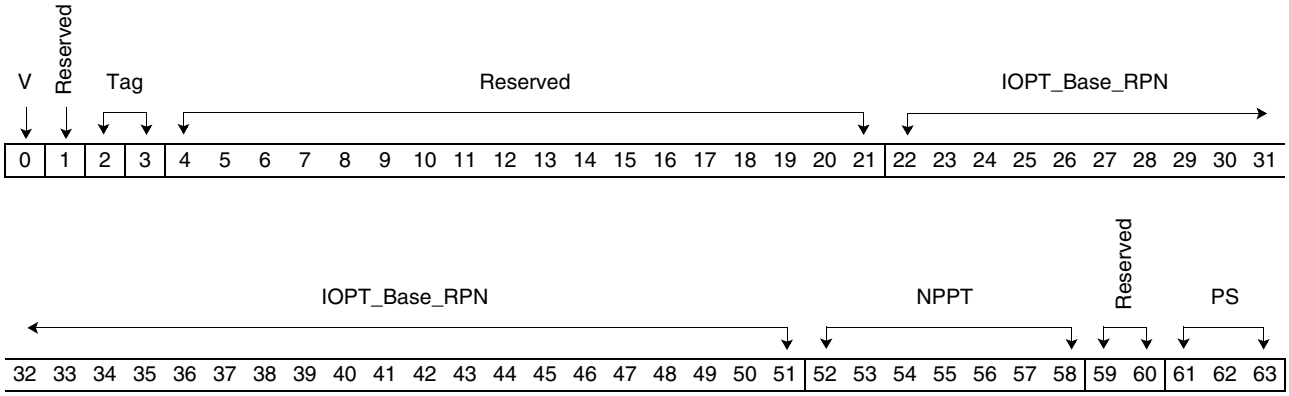
Register Short Name	IOC_IOPT_CacheDir	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘510000’ – x‘5101F8’: First way x‘510200’ – x‘5103F8’: Second way x‘510400’ – x‘5105F8’: Third way x‘510600’ – x‘5107F8’: Fourth way	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC

Bit(s)	Field Name	Description
0:29	Reserved	Bits are not implemented; all bits read back zero.
30	V	Valid 0 The IOPT cache entry is not valid. 1 The IOPT cache entry is valid.
31:63	Tag	IOPT directory tag.

5.2 IOC IOST Cache Register (IOC_IOST_Cache)

The IOST Cache register is a direct-mapped cache with 32 entries. From an MMIO standpoint, the IOST cache directory is considered to be part of the IOST cache. The fields in the IOST cache register have the same meaning as the corresponding fields in the IOST cache. However, there is no hint bit in the register. Only 30 bits of the IOPT Base real page number (RPN) are implemented. Also, the register V and Tag fields are fields in the IOST cache directory, so there are no corresponding fields in the IOST cache.

Register Short Name	IOC_IOST_Cache	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'510800' – x'5108F8'	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC

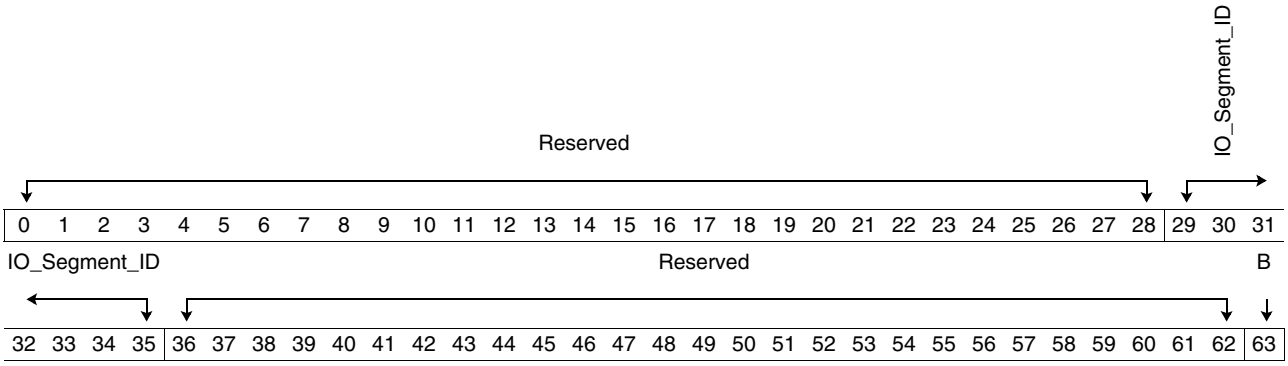


Bit(s)	Field Name	Description
0	V	Valid 0 IOST cache entry is not valid. 1 IOST cache entry is valid.
1	Reserved	Bits are not implemented; all bits read back zero.
2:3	Tag	IOST directory tag
4:21	Reserved	Bits are not implemented; all bits read back zero.
22:51	IOPT_Base_RPN	I/O Page Table Base Real Page Number This is the 4-KB RPN of the first entry in the I/O page table for this I/O segment.
52:58	NPPT	Number of 4-KB pages in the IOPT for this I/O segment minus one For IOPT format 1, the number of IOPT entries for the segment is equal to 512 times the sum of the NPPT plus 1. The IOPT entries are for the lowest-numbered pages in the segment. For IOPT format 2, the number of IOPT entries for the segment is equal to 256 times the sum of the NPPT plus 1.
59:60	Reserved	Bits are not implemented; all bits read back zero.
61:63	PS	Page Size of the RPN in the IOPT Entry That is, this field describes the page size of the pages in this I/O segment. The page size is 4^{PS} KB. Only page sizes of 4 KB, 64 KB, 1 MB, and 16 MB are supported. PS values of 1, 3, 5, and 7 are supported. All four page sizes can be used, regardless of the sizes used by the PowerPC Processor Element (PPE) and Synergistic Processor Element (SPE).

5.3 IOC IOST Cache Invalidate Register (IOC_IOST_CacheInvd)

This register allows software to invalidate the I/O segment table cache.

Register Short Name	IOC_IOST_CacheInvd	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'510900'	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC

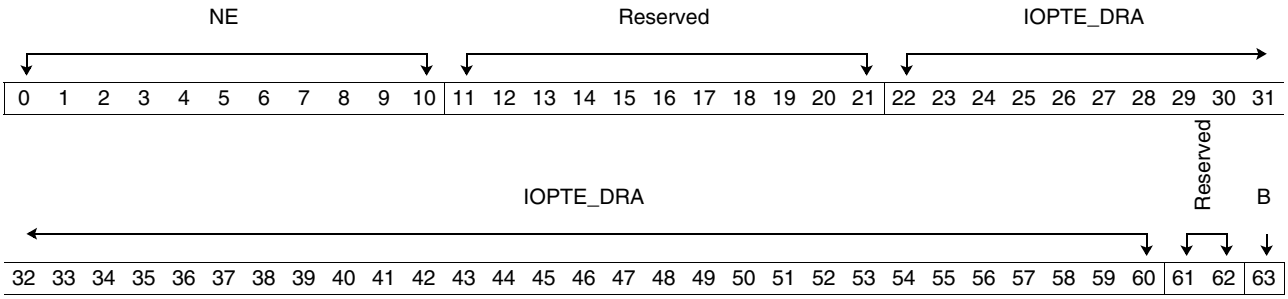


Bit(s)	Field Name	Description
0:28	Reserved	Bits are not implemented; all bits read back zero.
29:35	IO_Segment_ID	This is the I/O Segment ID to be invalidated in the IOST cache.
36:62	Reserved	Bits are not implemented; all bits read back zero.
63	B	Busy 0 The IOST Segment Invalidate operation is complete. 1 The IOST Segment Invalidate operation is in progress.

5.4 IOC IOPT Cache Invalidate Register (IOC_IOPT_CacheInvd)

This register allows software to invalidate entries in the I/O page table cache that correspond to the specified IOPT entries.

Register Short Name	IOC_IOPT_CacheInvd	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘510908’	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	Xlate

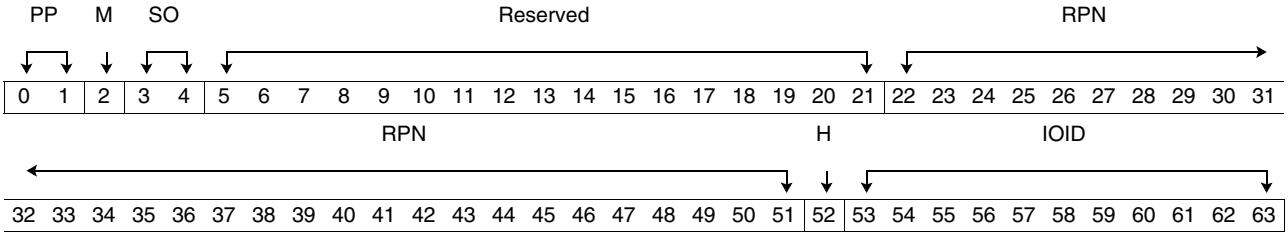


Bit(s)	Field Name	Description
0:10	NE	Number of Entries. The value in this field is one less than the number of entries in the IOPT for which IOPT cache entries are to be invalidated.
11:21	Reserved	Bits are not implemented; all bits read back zero.
22:60	IOPTE_DRA	I/O Page Table Entry Doubleword Real Address (IOPTE_DRA concatenated with ‘000’) is the real address of the first IOPT entry to be invalidated. All entries at or between (IOPTE_DRA concatenated with ‘000’) and ((IOPTE_DRA + NE) concatenated with ‘000’) are invalidated.
61:62	Reserved	Bits are not implemented; all bits read back zero.
63	B	Busy 0 The IOPT Cache Invalidate operation is complete. 1 The IOPT Cache Invalidate operation is in progress.

5.5 IOC IOPT Cache Register (IOC_IOPT_Cache)

The fields in the IOPT cache register have the same meaning as the corresponding fields in the IOPT. However, only 30 bits of RPN are implemented. When the IOPT cache directory entry is written, the contents of the IOPT cache register is moved into the corresponding IOPT cache entry. When an IOPT cache directory entry is read, the IOPT cache register is loaded from the corresponding IOPT cache entry.

Register Short Name	IOC_IOPT_Cache	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘510910’	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC

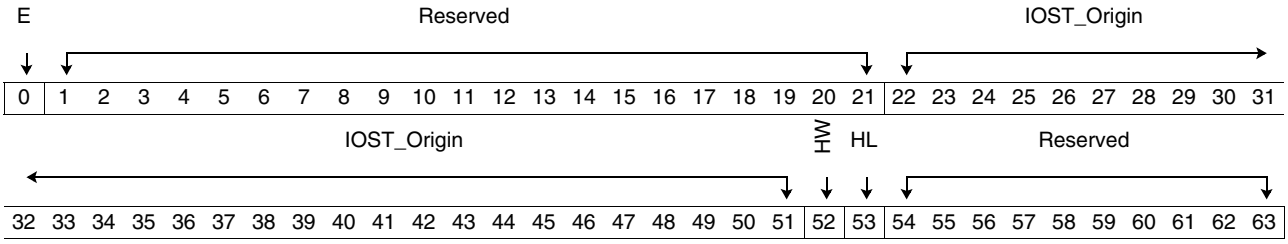


Bit(s)	Field Name	Description
0:1	PP	Page Protection 00 No access 01 Read access 10 Write access 11 Read and write access
2	M	Coherency Required 0 Not required 1 Required
3:4	SO	Storage Ordering 00 Ordered only if the I/O Device Identifier (IOID), IO Virtual Channel (VC), and IO address match. 01 Ordered only if the IOID, IO VC, and IO address match. 10 Ordered only if the IOID, IO VC, and IO address match, or IOID and VC match and the previous command is a write. 11 Ordered if the IOID and VC match.
5:21	Reserved	Bits are not implemented; all bits read back zero.
22:51	RPN	Real Page Number of the translated I/O address
52	H	Hint enable
53:63	IOID	I/O Device Identifier. Only the I/O device specified can have its I/O address translated using this IOPT entry.

5.6 IOC I/O Segment Table (IOST) Origin Register (IOC_IOST_Origin)

This register sets up the I/O translation mechanism in the IOC. The IOST Size field is not implemented. When I/O translation is enabled and hardware miss handling is enabled, the IOST is treated as having a fixed size with 128 entries.

Register Short Name	IOC_IOST_Origin	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘510918’	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC



Bit(s)	Field Name	Description
0	E	I/O Translation Enable. The IOC_IOCmd_Cfg[TE] must be set to the same value as this E bit. 0 Disabled. I/O addresses are treated as real addresses. The I/O address translation unit is disabled to save power. This is the only IOC Address Translation MMIO register that can be accessed. 1 Enabled. I/O addresses are translated using the IOST and IOPT(s).
1:21	Reserved	Bits are not implemented; all bits read back zero.
22:51	IOST_Origin	The Real Page Number (RPN) of the IOST. The IOST must be integrally aligned and must be aligned on a 4-KB page.
52	HW	Enable Hardware Miss Handling for IOST and IOPT caches.
53	HL	Hint Lock
54:63	Reserved	Bits are not implemented; all bits read back zero.

5.7 IOC I/O Exception Status Register (IOC_IO_ExcpStat)

When an I/O exception occurs, debug information is captured in this register, and then IOC_IO_ExcpStat[0] is set to '1'. Debug information for an I/O exception is only captured when IOC_IO_ExcpStat[0] is set to '0'. After handling an I/O exception, software should set IOC_IO_ExcpStat[0] to '0' to enable the capturing of information on a subsequent I/O exception.

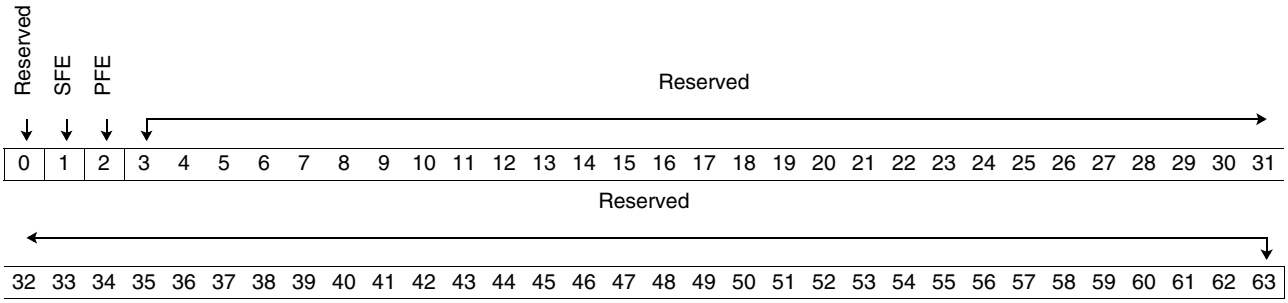
Register Short Name	IOC_IO_ExcpStat	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'510920'	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC

Bit(s)	Field Name	Description
0	V	Valid. Software must set this bit to '0' after handling the exception. 0 Errors that occur when V is set to '0' are not captured. 1 The I/O Exception Status register contains error status for the first error detected.
1:2	SPF	Segment or Page Fault Exception The following values assume that bits [1:2] were set to '00' before the exception occurred. 00 No I/O address translation fault occurred. 01 An I/O page fault occurred. 10 Undefined 11 An I/O segment fault occurred.
3:28	Reserved	Bits are not implemented; all bits read back zero.
29:51	Address	Bits [7:29] of the 42-bit I/O address are used for the access that caused the I/O exception.
52	RW	Read or write type of I/O access 0 Write 1 Read
53:63	IOID	I/O Device Identifier

5.8 IOC I/O Exception Mask Register (IOC_IO_ExcpMask)

This register configures the mask for interrupts due to I/O exceptions. These masks do not affect the setting of the I/O Exception Status Register (IOC_IO_ExcpStat).

Register Short Name	IOC_IO_ExcpMask	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'510928'	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC

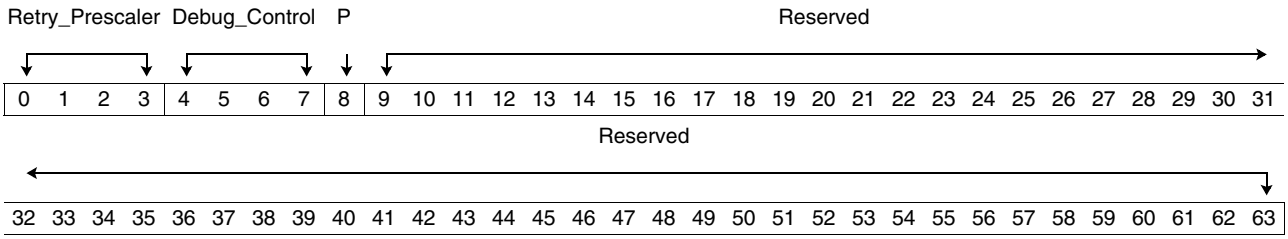


Bit(s)	Field Name	Description
0	Reserved	Bits are not implemented; all bits read back zero.
1	SFE	I/O segment fault mask exception enable
2	PFE	I/O page fault mask exception enable
3:63	Reserved	Bits are not implemented; all bits read back zero.

5.9 IOC Translation Configuration Register (IOC_XlateCfg)

This register configures the retry backoff prescaler and power management of the I/O translation (Xlate) hardware.

Register Short Name	IOC_XlateCfg	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘510930’	**Memory Map Area**	IOC Address Translation
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IOC



Bit(s)	Field Name	Description
0:3	Retry_Prescaler	Prescaler for the decrement value used to determine the amount of time before re-issuing an element interconnect bus (EIB) read after a Retry combined response. The first 2 bits of this field are used to select which decrement of the Linear Feedback Shift Register (LFSR) backoff to apply. The next 2 bits are used to select the LFSR sequence length. **Encode** **Decrement Rate** **LFSR Sequence Length** ‘00x’ 1:1 ‘01x’ 1:2 ‘10x’ 1:4 ‘11x’ 1:8 ‘x00’ 281 ‘x01’ 317 ‘x10’ 439 ‘x11’ 487
4:7	Debug_Control	Debug control. Nonzero values indicate conditions that halt the I/O translation (Xlate) hardware. 0000 Never halt 0001 I/O Segment Fault or I/O Page Fault occurs 0010 I/O Address Translation Exception interrupt is signalled to the internal interrupt controller (IIC). 0011 Write to the IOPT Cache Invalidate register that invalidates way 0 0100 Write to the IOPT Cache Invalidate register that invalidates way 1 0101 Write to the IOPT Cache Invalidate register that invalidates way 2 0110 Write to IOPT Cache Invalidate register that invalidates way 3 0111 Write to the IOST Cache 1000 Write to the IOPT Cache Directory for the first way 1001 Write to the IOPT Cache Directory for the second way 1010 Write to the IOPT Cache Directory for the third way 1011 Write to theIOPT Cache Directory for the fourth way 1100 IOST Cache miss or IOPT Cache miss occurs on I/O translation 1101 Xlate hardware issues a “clear” to XIN logic All values not shown above are invalid.

Bit(s)	Field Name	Description
8	P	Power Management Control for I/O translation (Xlate) hardware. 0 Enable latches. 1 Disable latches. When this bit is disabled, all latches in Xlate are disabled. The Xlate hardware can be reenabled by asserting the HARD_RESET# signal to reset the Cell Broadband Engine.
9:63	Reserved	Bits are not implemented; all bits read back zero.

6. Internal Interrupt Controller (IIC) MMIO Registers

This section describes the IIC memory-mapped I/O (MMIO) registers. *Table 6-1* shows the IOC address translation MMIO memory map and lists the IOC registers. The Internal Interrupt Controller (IIC) space starts at x‘508 000’ and ends at x‘508 FFF’. Offsets are from the start of the IIC register space. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

6.1 IIC MMIO Memory Map

Table 6-1. IIC Memory Map

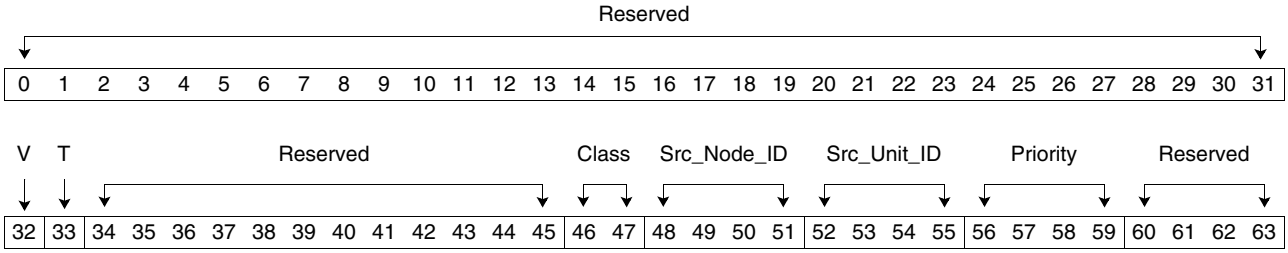
Hexadecimal Offset (x‘508 *nnn*’)	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
x‘000’ – x‘3FF’	Reserved			
x‘400’	*IIC Thread 0 Interrupt Pending Port Register (IIC_IPP0)* (non-destructive)	64	R	*Section 6.2.1* on page 102
x‘408’	*IIC Thread 0 Interrupt Pending Port Register (IIC_IPP0)* (destructive)	64	R	*Section 6.2.1* on page 102
x‘410’	*IIC Thread 0 Interrupt Generation Port Register (IIC_IGP0)*	64	W	*Section 6.2.2* on page 103
x‘418’	*IIC Thread 0 Current Priority Level Register (IIC_CPL0)*	64	R/W	*Section 6.2.3* on page 104
x‘420’	*IIC Thread 1 Interrupt Pending Port Register (IIC_IPP1)* (non-destructive)	64	R	*Section 6.2.4* on page 105
x‘428’	*IIC Thread 1 Interrupt Pending Port Register (IIC_IPP1)* (destructive)	64	R	*Section 6.2.4* on page 105
x‘430’	*IIC Thread 1 Interrupt Generation Port Register (IIC_IGP1)*	64	W	*Section 6.2.5* on page 106
x‘438’	*IIC Thread 1 Current Priority Level Register (IIC_CPL1)*	64	R/W	*Section 6.2.6* on page 107
x‘440’	*IIC Interrupt Routing Register (IIC_IR)*	64	R/W	*Section 6.2.7* on page 108
x‘448’	*IIC_Interrupt Status Register (IIC_IS)*	64	R/W	*Section 6.2.8* on page 109
x‘450’ – x‘FFF’	Reserved			

6.2 IIC Register Descriptions

6.2.1 IIC Thread 0 Interrupt Pending Port Register (IIC_IPP0)

The IIC_IPP0 register allows software to read the interrupt source and other information about any pending interrupts. After software reads the pending port, the next highest interrupt is loaded into the IPP. When reading the IPP nondestructively, the value of the associated Current Priority Level (CPL) register (IIC_CPL0) is not updated with the interrupt in the IPP. When reading the IPP destructively, the CPL register takes on the priority of the interrupt in the IPP. The destructive read address offset is always 8 bytes beyond the non-destructive read offset.

Register Short Name	IIC_IPP0	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘508400’ Nondestructive x‘508408’ Destructive	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC



Bits	Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32	V	Interrupt Valid 0 No interrupt pending. 1 Interrupt pending.
33	T	Interrupt Type 0 Interrupt from Synergistic Processor Element (SPE), external device, or external interrupt controller 1 Interrupt from thread 0 Interrupt Generation Port
34:45	Reserved	Bits are not implemented; all bits read back zero.
46:47	Class	Interrupt Class. Returns zeros when T = ‘1’.
48:51	Src_Node_ID	Interrupt Source Cell Broadband Engine Interface (BIF) Node ID. Returns zeros when T = ‘1’.
52:55	Src_Unit_ID	Interrupt Source Unit ID. Returns zeros when T = ‘1’.
56:59	Priority	Interrupt Priority
60:63	Reserved	Bits are not implemented; all bits read back zero.

6.2.2 IIC Thread 0 Interrupt Generation Port Register (IIC_IGP0)

The IIC_IGP0 register allows software to generate interrupts to PowerPC Processor Element (PPE) thread 0.

Register Short Name	IIC_IGP0	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'508410'	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC

Bit(s)	Field Name	Description
0:55	Reserved	Bits are not implemented; all bits read back zero.
56:59	Priority	Interrupt Priority
60:63	Reserved	Bits are not implemented; all bits read back zero.

6.2.3 IIC Thread 0 Current Priority Level Register (IIC_CPL0)

The IIC_CPL0 register allows software to mask interrupts of a specified priority. Software directly loads the CPL using an MMIO write of the register or indirectly using a destructive read of the IPP. When executing a destructive read of the IPP, the priority of the interrupt in the IPP is written to the CPL if that interrupt is valid. Otherwise, the content is unchanged.

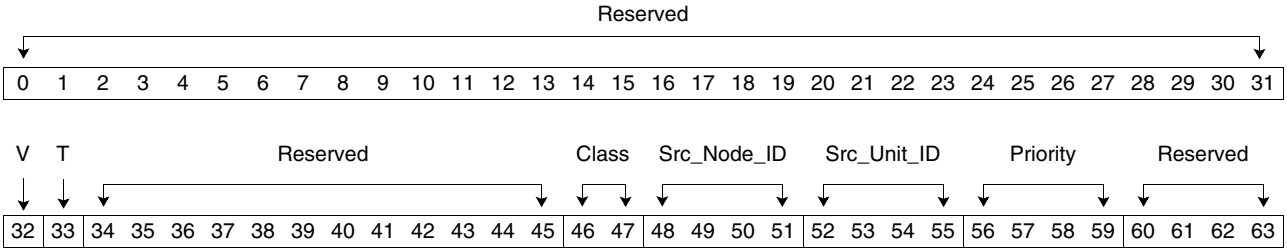
Register Short Name	IIC_CPL0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'508418'	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC

Bit(s)	Field Name	Description
0:55	Reserved	Bits are not implemented; all bits read back zero.
56:60	Priority	If the priority of the interrupt in the IIC_IPP0 register is numerically less than the priority in this register, then the thread 0 external interrupt signal to the PPU is active.
61:63	Reserved	Bits are not implemented; all bits read back zero.

6.2.4 IIC Thread 1 Interrupt Pending Port Register (IIC_IPP1)

The IIC_IPP1 register allows software to read the interrupt source and other information about any pending interrupts. After software reads the pending port, the next highest interrupt is loaded into the IPP. When reading the IPP nondestructively, the value of the associated Current Priority Level (CPL) register (IIC_CPL1) is not updated with the interrupt in the IPP. When reading the IPP destructively, the CPL register takes on the priority of the interrupt in the IPP. The destructive read address offset is always 8 bytes beyond the non-destructive read offset.

Register Short Name	IIC_IPP1	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘508420’ Nondestructive x‘508428’ Destructive	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC



Bits	Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32	V	Interrupt Valid 0 No interrupt pending 1 Interrupt pending
33	T	Interrupt Type 0 Interrupt from Synergistic Processor Element (SPE), external device, or external interrupt controller 1 Interrupt from thread 1 Interrupt Generation Port
34:45	Reserved	Bits are not implemented; all bits read back zero.
46:47	Class	Interrupt Class. Returns zeros when T = ‘1’.
48:51	Src_Node_ID	Interrupt Source Cell Broadband Engine Interface (BIF) Node ID. Returns zeros when T = ‘1’.
52:55	Src_Unit_ID	Interrupt Source Unit ID. Returns zeros when T = ‘1’.
56:59	Priority	Interrupt Priority
60:63	Reserved	Bits are not implemented; all bits read back zero.

6.2.5 IIC Thread 1 Interrupt Generation Port Register (IIC_IGP1)

The IIC_IGP1 register allows software to generate interrupts to PowerPC Processor Element (PPE) thread 1.

Register Short Name	IIC_IGP1	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'508430'	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC

Bit(s)	Field Name	Description
0:55	Reserved	Bits are not implemented; all bits read back zero.
56:59	Priority	Interrupt Priority
60:63	Reserved	Bits are not implemented; all bits read back zero.

6.2.6 IIC Thread 1 Current Priority Level Register (IIC_CPL1)

The IIC_CPL1 register allows software to mask interrupts of a specified priority. Software directly loads the CPL using an MMIO write of the register or indirectly using a destructive read of the IPP. When executing a destructive read of the IPP, the priority of the interrupt in the IPP is written to the CPL if that interrupt is valid. Otherwise, the content is unchanged.

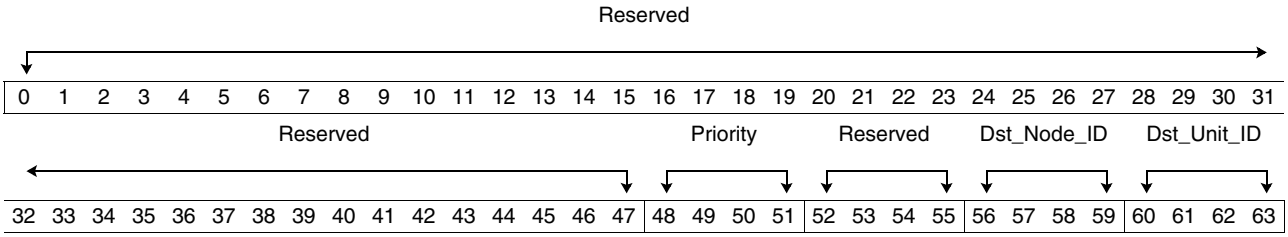
Register Short Name	IIC_CPL1	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'508438'	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC

Bit(s)	Field Name	Description
0:55	Reserved	Bits are not implemented; all bits read back zero.
56:60	Current_Priority	If the priority of the interrupt in the IIC_IPP1 is numerically less than the priority in this register, then the thread 1 external interrupt signal to the PPU is active.
61:63	Reserved	Bits are not implemented; all bits read back zero.

6.2.7 IIC Interrupt Routing Register (IIC_IR)

The IIC_IR register configures the priority and destination of interrupts in the IIC_IS register.

Register Short Name	IIC_IR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘508440’	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC

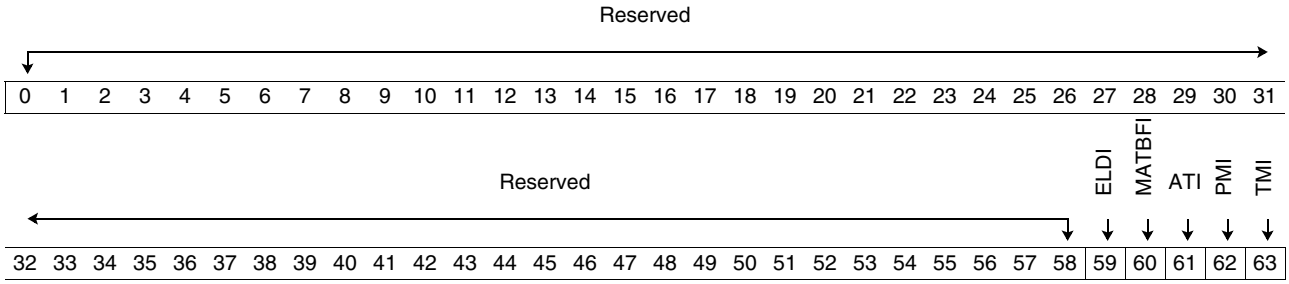


Bit(s)	Field Name	Description
0:47	Reserved	Bits are not implemented; all bits read back zero.
48:51	Priority	Interrupt Priority. This field corresponds to the 4 bits of the priority field in the IIC_IPP0 and IIC_IPP1.
52:55	Reserved	Bits are not implemented; all bits read back zero.
56:59	Dst_Node_ID	Destination Node BIF ID for interrupts in the IIC_IS register.
60:63	Dst_Unit_ID	Destination Unit ID for interrupts in the IIC_IS register. Interrupt packets routed to I/O Interface 0 (IOIF0) have the same destination unit ID as I/O Controller 0 (IOC0). Those that are routed to I/O Interface 1 (IOIF1) have the same destination unit ID as IOC1. The valid values are: 0000 IOC0 1011 I/O Controller 1 (IOC1) 1110 Processor thread 0 1111 Processor thread 1

6.2.8 IIC_Interrupt Status Register (IIC_IS)

The IIC_IS register records interrupt conditions from the Memory Interface Controller (MIC), Element Interconnect Bus (EIB) unit, I/O Address Translation (Xlate) , Performance Monitoring (PM), and Token Manager (TKM). When an interrupt occurs, the corresponding bit is set to '1'. If the Interrupt Status Register is nonzero, the IIC creates a class 1 interrupt that goes to either the Element Interrupt Bus (EIB), IPP0, or IPP1 depending on how the IIC_IR is configured. After resetting the interrupt condition in the source unit, software resets the interrupt by writing '1' to the corresponding bit position in the Interrupt Status Register. Writing '0' to the corresponding bit has no effect on that bit. Writing this register must occur to confirm handling of interrupts in the Interrupt Status Register.

Register Short Name	IIC_IS	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'508448'	**Memory Map Area**	Internal Interrupt Controller (IIC)
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IIC



Bit(s)	Field Name	Description
0:58	Reserved	All bits read back zero.
59	ELDI	EIB Possible Livelock Detection Interrupt 0 Inactive 1 Active
60	MATBFI	MIC Auxiliary Trace Buffer Full Interrupt 0 Inactive 1 Active
61	ATI	I/O Address Translation Interrupt Active
62	PMI	Performance Monitor Interrupt Active
63	TMI	Token Manager Interrupt Active

7. Token Manager (TKM) MMIO Registers

This section shows the TKM memory map and lists the TKM registers. The shared MIC and TKM memory space starts at x‘50A 000’ and ends at x‘50A FFF’. Offsets are from the start of the shared MIC and TKM register space. For an overview of the complete CBE memory map, see *Table 1-3* on page 14.

This section describes the TKM memory-mapped I/O (MMIO) registers. *Table 7-1* shows the BEI IOC MMIO memory map and lists the BEI IOC registers. The shared MIC and TKM memory space starts at x‘50A 000’ and ends at x‘50A FFF’. Offsets are from the start of the shared MIC and TKM register space. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register bit definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

7.1 TKM MMIO Memory Map

Table 7-1. TKM MMIO Memory Map

Hexadecimal Offset (x‘50A *nnn*’)	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
x‘FC0’ – x‘FE0’	Reserved			
x‘FE8’	*TKM Interrupt Status Register (TKM_IS)*	64	R/W	*Section 7.2.1* on page 112
x‘FF0’	Reserved			

7.2 TKM MMIO Register Descriptions

7.2.1 TKM Interrupt Status Register (TKM_IS)

The TKM_IS indicates which TKM events caused an external interrupt condition. If feedback for a managed resource indicates that the command queue is full, the corresponding TKM_IS register bit is set. When a TKM_IS register bit is set to ‘1’ and the corresponding interrupt is enabled, an exception signal from the TKM to the IIC is active. When this signal is active, the IIC sets the TKM exception bit in the IIC_IS register to ‘1’.

Reads of the TKM_IS register are nondestructive. Once a TKM_IS register bit is set to ‘1’, it remains set until software resets the bit. To reset TKM_IS register bit, software must write a binary one to the corresponding bit.

Register Short Name	TKM_IS	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘50AFE8’	**Memory Map Area**	MIC and TKM
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	TKM

Bit(s)	Field Name	Description
0:58	Reserved	Bits are not implemented; all bits read back zero.
59	M	MIC feedback Set to ‘1’ if MIC feedback indicates that the command queue is full.
60	I0	In IOIF0 feedback Set to ‘1’ if IOIF0 In feedback indicates that the command queue is full.
61	O0	Out IOIF0 feedback Set to ‘1’ if IOIF0 Out feedback indicates that the command queue is full.
62	I1	In IOIF1 feedback Set to ‘1’ if IOIF1 In feedback indicates that the command queue is full.
63	O1	Out IOIF1 feedback Set to ‘1’ if IOIF1 Out feedback indicates that the command queue is full.

8. Element Interconnect Bus (EIB) MMIO Registers

8.1 EIB MMIO Memory Map

Table 8-1 shows the EIB MMIO memory map and lists the EIB registers. The EIB register space starts at x‘511 800’ and ends at x‘511 BFF’. Offsets are from the start of the EIB register space. For an overview of the complete CBE memory map, see *Table 1-3* on page 14.

This section describes the EIB IOC memory-mapped I/O (MMIO) registers. *Table 8-1* shows the EIB MMIO memory map and lists the EIB registers. The EIB register space starts at x‘511 800’ and ends at x‘511 BFF’. Offsets are from the start of the EIB register space. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register bit definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

Table 8-1. EIB MMIO Memory Map

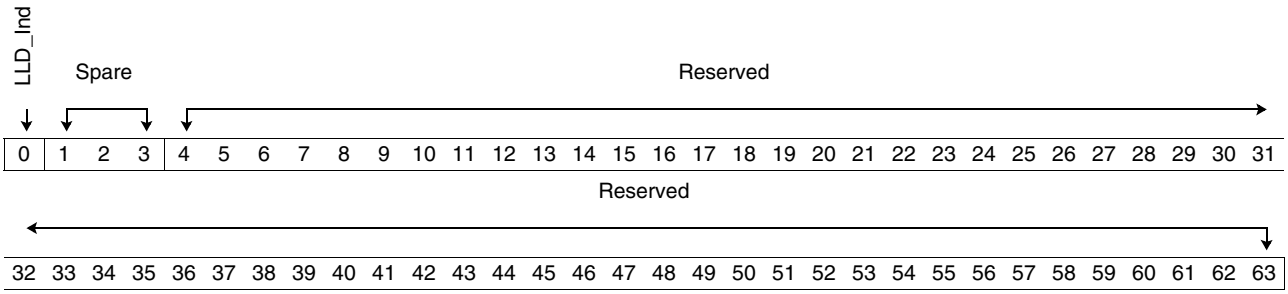
Hex Offset from BE_MMIO_Base (x‘511 *nnn*’)	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
x‘800’ – x‘808’	Reserved			
x‘810’	*EIB Interrupt Register (EIB_Int)*	64	R/W	*Section 8.2.1* on page 114
x‘840’	*EIB Local Base Address Register 0 (EIB_LBAR0)*	64	R/W	*Section 8.2.2* on page 115
x‘848’	*EIB Local Base Address Mask Register 0 (EIB_LBAMR0)*	64	R/W	*Section 8.2.3* on page 116
x‘850’	*EIB Local Base Address Register 1 (EIB_LBAR1)*	64	R/W	*Section 8.2.4* on page 117
x‘858’	*EIB Local Base Address Mask Register 1 (EIB_LBAMR1)*	64	R/W	*Section 8.2.5* on page 118
x‘860’ – x‘BFF’	Reserved			

8.2 EIB MMIO Register Descriptions

8.2.1 EIB Interrupt Register (EIB_Int)

This register is used to log that a possible livelock condition has been detected on the EIB.

Register Short Name	EIB_Int	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511810'	**Memory Map Area**	EIB
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	EIB



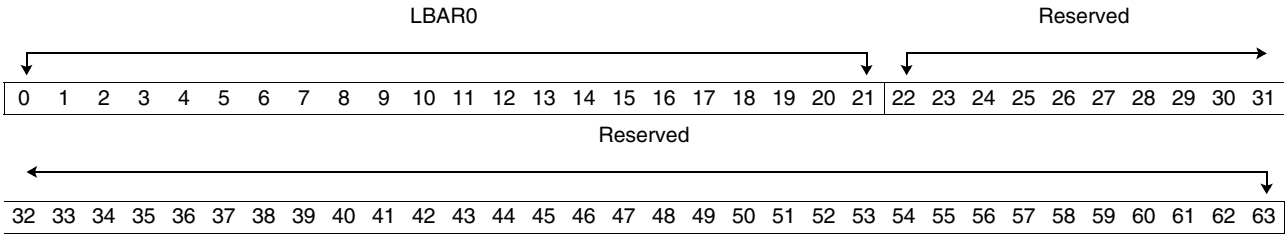
Bits	Field Name	Description
0	LLD_Ind	EIB Possible LiveLock Detection Indicator 0 None detected 1 A possible livelock condition (overuse of the previous adjacent address match (PAAM) waitbuffer) has been detected. See the "EIB Possible Livelock Detection Interrupt" in the *Cell Broadband Engine Book IV*. See your IBM representative for access to this confidential document. **Note:** An MMIO write that sets this bit to '1' when it was previously '0' causes IIC_IS[59] to be set, provided EIB_AC0_CTL[16] is '0', and regardless of the state of EIB_AC0_CTL[13:15].
1:3	Spare	Spare Bits This field can be read from or written to without side effects.
4:63	Reserved	Bits are not implemented; all bits read back zero.

8.2.2 EIB Local Base Address Register 0 (EIB_LBAR0)

This register contains the base address of the primary noncoherent address range. This base address is used in conjunction with the EIB_LBAMR0 register to determine whether a command address falls within noncoherent range 0.

Read and write commands in this address range are considered to be local commands. Local commands have the M bit set to '0' and are reflected back to bus masters after arbitration.

Register Short Name	EIB_LBAR0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511840'	**Memory Map Area**	EIB
Value at Initial POR	N/A	**Value During POR Set By**	Configuration ring
Specification Type	Implementation-specific register	**Unit**	EIB



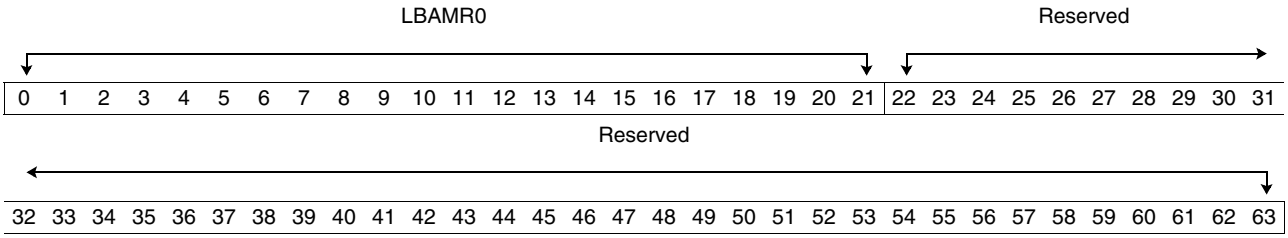
Bits	Field Name	Description
0:21	LBAR0	Defines the primary noncoherent command address range. Bits [0:21] of this register correspond to bits [22:43] of the Real Address used on the EIB.
22:63	Reserved	Bits are not implemented; all bits read back zero.

8.2.3 EIB Local Base Address Mask Register 0 (EIB_LBAMR0)

This register is used to hold the mask (bit enables) for the primary noncoherent address range. This mask is used to specify which bits from the command address should be compared to the EIB_LBAR0 register to determine whether the command address falls within noncoherent range 0.

Read and write commands in the noncoherent range are considered to be local commands. Local commands have the M bit set to '0' and are reflected back to bus masters after arbitration.

Register Short Name	EIB_LBAMR0	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511848'	**Memory Map Area**	EIB
Value at Initial POR	N/A	**Value During POR Set By**	Configuration ring
Specification Type	Implementation-specific register	**Unit**	EIB



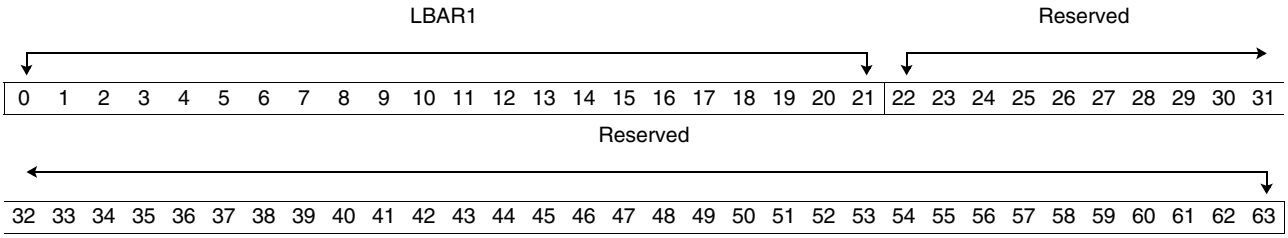
Bits	Field Name	Description
0:21	LBAMR0	Local Base Address Mask Register 0 Mask for use with Local Base Address Register 0
22:63	Reserved	Bits are not implemented; all bits read back zero.

8.2.4 EIB Local Base Address Register 1 (EIB_LBAR1)

This register contains the base address of the secondary noncoherent address range. This base address is used in conjunction with the EIB_LBAMR1 register to determine if a command address falls within noncoherent range 1.

Read and write commands in this address range are considered to be local commands. Local commands have the M bit set to '0' and are reflected back to bus masters after arbitration.

Register Short Name	EIB_LBAR1	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511850'	**Memory Map Area**	EIB
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	EIB



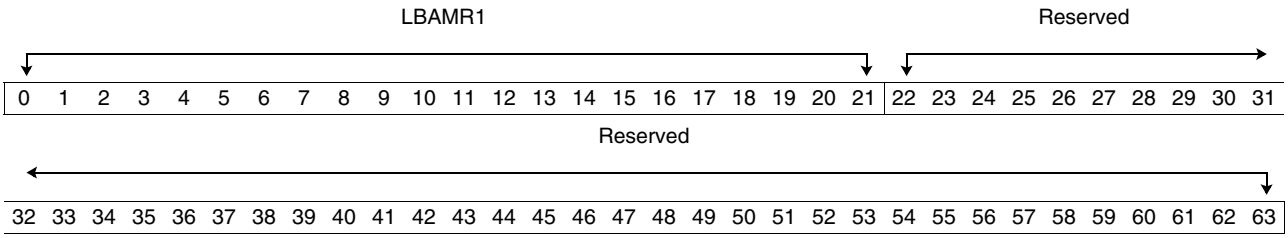
Bits	Field Name	Description
0:21	LBAR1	Local Base Address Register 1 Defines secondary noncoherent command address range. Bits [0:21] of this register correspond to bits [22:43] of the Real Address used on the EIB.
22:63	Reserved	Bits are not implemented; all bits read back zero.

8.2.5 EIB Local Base Address Mask Register 1 (EIB_LBAMR1)

This register contains the mask (bit enables) for the secondary noncoherent address range. This mask is used to specify which bits from the command address should be compared to the EIB_LBAR1 register to determine if the command address falls within noncoherent range 1.

Read and write commands in the noncoherent range are considered to be local commands. Local commands have the M bit set to '0' and are reflected back to bus masters after arbitration.

Register Short Name	EIB_LBAMR1	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'511858'	**Memory Map Area**	EIB
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	EIB



Bits	Field Name	Description
0:21	LBAMR1	Local Base Address Mask Register 1 Mask for use with Local Base Address Register 1
22:63	Reserved	Bits are not implemented; all bits read back zero.

9. Pervasive MMIO Registers

This section describes the Cell Broadband Engine Interface Pervasive memory-mapped I/O (MMIO) registers. *Table 9-1* shows the Pervasive Power and Thermal Management MMIO memory map and lists the Pervasive registers. The Pervasive MMIO Registers area starts at x‘509 000’ and ends at x‘509 FFF’. Offsets are from the start of the Pervasive register space. For the complete CBE MMIO memory map, see *Section 1 Cell Broadband Engine Memory-Mapped I/O Registers* on page 13.

Notes on the register definitions:

- Registers that have no implementation-specific bit definitions (all bits are as defined in the architecture) and have initial power-on reset (POR) values scanned to all zeros show only a cross-reference to *Appendix A Registers Defined in the CBEA* on page 181.
- Multiple address offsets for a register indicate that there are multiple instances of this register.
- The *Privilege Type* of all MMIO registers is recommended by the *Cell Broadband Engine Architecture*, but is not enforced in hardware.
- The *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.

9.1 Pervasive MMIO Registers

Within the (RAS) TCU shared memory space, the 32-bit registers are defined on even word addresses. This means that these 32-bit registers are defined on bits [0:31] of the MMIO doubleword.

Table 9-1. Pervasive Registers (Page 1 of 2)

Hexadecimal Offset (x‘509000’ - x‘509FFF’	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
Performance Monitor Registers Shared with the TLA				
x‘C00’ – x‘008’	Reserved			
x‘108’	*Trace Buffer High Doubleword Register (0 to 63) (trace_buffer_high)*	64	R	*Section 9.2.1* on page 121
x‘110’	*Trace Buffer Low Doubleword Register (64 to 127) (trace_buffer_low)*	64	R	*Section 9.2.2* on page 122
x‘118’	*Trace Address Register (trace_address)*	64	R/W	*Section 9.2.3* on page 123
x‘120’	*External Trace Timer Register (ext_tr_timer)*	64	W	*Section 9.2.4* on page 124
Performance Monitor Only Registers				
x‘400’	*Performance Monitor Status/Interrupt Mask Register (pm_status)*	32	R/W	*Section 9.3.1* on page 125
x‘408’	*Performance Monitor Control Register (pm_control)*	32	W	*Section 9.3.2* on page 126
x‘410’	*Performance Monitor Interval Register (pm_interval)*	32	R/W	*Section 9.3.3* on page 129
x‘418’	*Performance Monitor Counter Pairs Registers (pmM_N)*	32	R/W	*Section 9.3.4* on page 130
x‘420’				
x‘428’				
x‘430’				

Table 9-1. Pervasive Registers (Page 2 of 2)

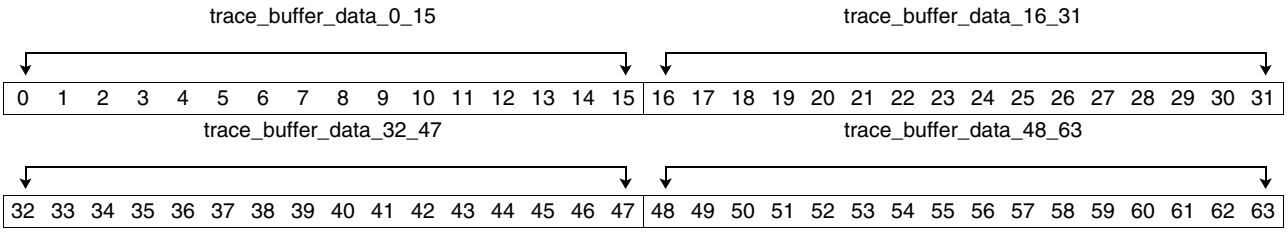
Hexadecimal Offset (x'509000' - x'509FFF'	Register Name and (Short Name)	Width (Bits)	Read/ Write	Additional Information
x'438'	*Performance Monitor Start Stop Register (pm_start_stop)*	32	W	*Section 9.3.5* on page 131
x'440'	*Performance Monitor Counter Control Registers (pmN_control)*	32	W	*Section 9.3.6* on page 133
x'448'				
x'450'				
x'458'				
x'460'				
x'468'				
x'470'				
x'478'				
Power Management Control Registers				
x'880'	*Power Management Control Register (PMCR)*	64	R/W	*Section 9.4.1* on page 135
x'888'	*Power Management Status Register (PMSR)*	64	R	*Section 9.4.2* on page 137
Thermal Management MMIO Registers				
x'800' – x'818'	Reserved			
x'820'	*Thermal Sensor Interrupt Temperature Register 1 (TS_ITR1)*	64	R/W	*Section 9.5.1* on page 139
x'828'	*Thermal Sensor Interrupt Temperature Register 2 (TS_ITR2)*	64	R/W	*Section 9.5.2* on page 141
x'830'	Reserved			
x'838'	*Thermal Sensor Interrupt Status Register (TS_ISR)*	64	R/W	*Section 9.5.3* on page 142
x'840'	*Thermal Sensor Interrupt Mask Register (TS_IMR)*	64	R/W	*Section 9.5.4* on page 143
x'848' – x'850'	Reserved			
x'858'	*Thermal Management System Interrupt Mask Register (TM_SIMR)*	64	R/W	*Section 9.5.5* on page 145
x'860' – x'890'	Reserved			

9.2 Performance Monitor Registers Shared with the Trace Logic Analyzer

9.2.1 Trace Buffer High Doubleword Register (0 to 63) (trace_buffer_high)

This register is shared with the TLA.

Register Short Name	trace_buffer_high	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509108'	**Memory Map Area**	Pervasive: Trace Logic Analyzer
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

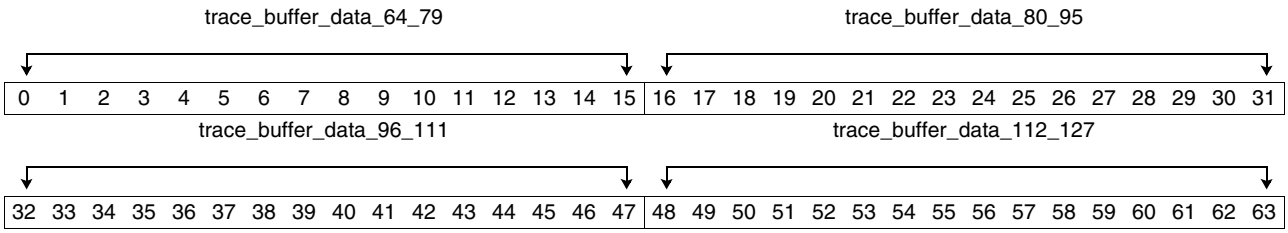


Bit(s)	Field Name	Description
0:15	trace_buffer_data_0_15	PM0 when configured as 16-bit counters; PM0[0:15] when configured as a 32-bit counter.
16:31	trace_buffer_data_16_31	PM4 when configured as 16-bit counters; PM0[16:31] when configured as a 32-bit counter.
32:47	trace_buffer_data_32_47	PM1 when configured as 16-bit counters; PM1[0:15] when configured as a 32-bit counter.
48:63	trace_buffer_data_48_63	PM5 when configured as 16-bit counters; PM1[16:31] when configured as a 32-bit counter.

9.2.2 Trace Buffer Low Doubleword Register (64 to 127) (trace_buffer_low)

This register is shared with the TLA. The read address is incremented on a read of this register, so this register must be read with a 64-bit MMIO read.

Register Short Name	trace_buffer_low	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509110'	**Memory Map Area**	Pervasive: Trace Logic Analyzer
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



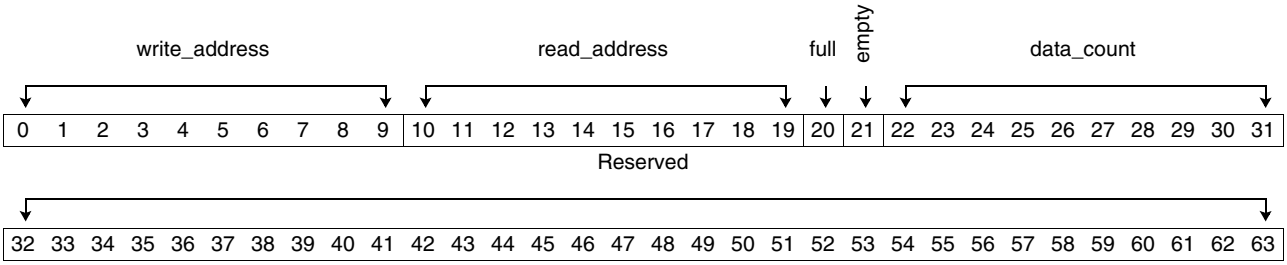
Bit(s)	Field Name	Description
0:15	trace_buffer_data_64_79	PM2 when configured as 16-bit counters; PM2[0:15] when configured as a 32-bit counter.
16:31	trace_buffer_data_80_95	PM6 when configured as 16-bit counters; PM2[16:31] when configured as a 32-bit counter.
32:47	trace_buffer_data_96_111	PM3 when configured as 16-bit counters; PM3[0:15] when configured as a 32-bit counter.
48:63	trace_buffer_data_112_127	PM7 when configured as 16-bit counters; PM3[16:31] when configured as a 32-bit counter.

9.2.3 Trace Address Register (trace_address)

This register is shared with the TLA.

The trace_address register must be written to zero before enabling the performance monitor in performance data trace modes. Since the performance-monitor logic controls the trace arrays as a hardware FIFO during operation, it is unnecessary and undesirable to write the trace address registers. When in the trace_buffer_overwrite mode, the write and read pointers wrap when the maximum count is reached. A read from the FIFO with no data in the FIFO returns invalid data and sets a trace buffer underflow interrupt condition. See *Section 9.3.1 Performance Monitor Status/Interrupt Mask Register (pm_status)* on page 125.

Register Short Name	trace_address	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509118'	**Memory Map Area**	Pervasive: Trace Logic Analyzer
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

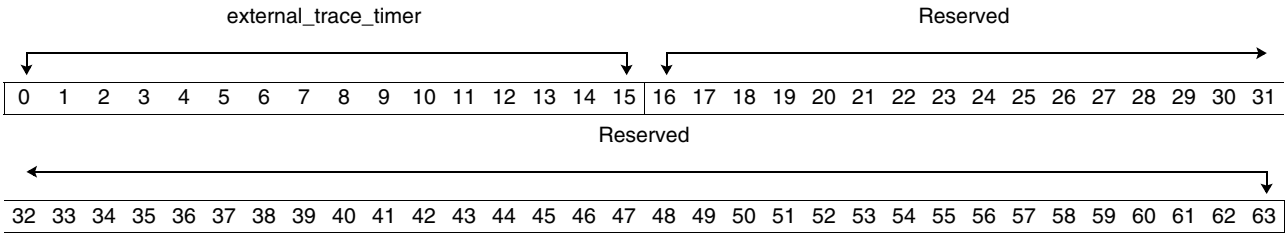


Bit(s)	Field Name	Description
0:9	write_address	Address for trace array writing.
10:19	read_address	Address for trace array reading.
20	full	Trace array full (read only)
21	empty	Trace array empty (read only)
22:31	data_count	Count of trace array addresses containing valid data (read only)
32:63	Reserved	Bits are not implemented; all bits read back zero

9.2.4 External Trace Timer Register (ext_tr_timer)

The external trace timer sets the upper limit for data to be transferred from the trace buffer.

Register Short Name	ext_tr_timer	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x‘509120’	**Memory Map Area**	Pervasive: Trace Logic Analyzer
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



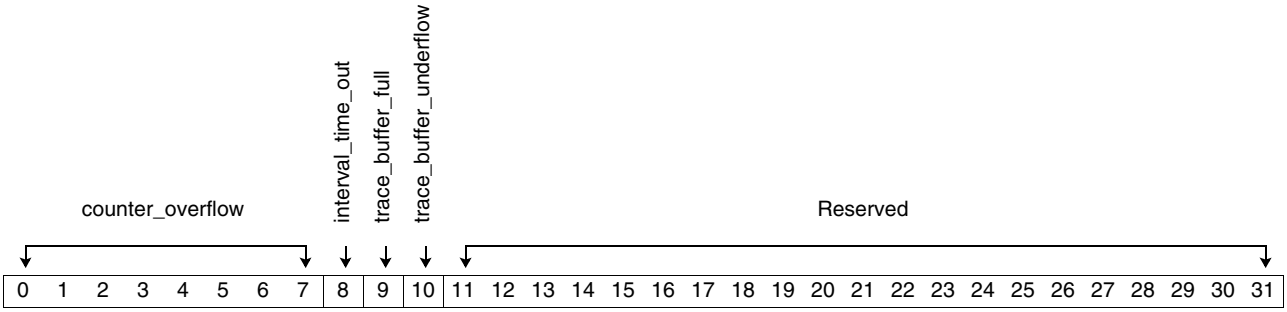
Bit(s)	Field Name	Description
0:15	external_trace_timer	The external trace timer sets the upper limit for data to be transferred from the trace buffer. These 16 bytes are burst over the auxiliary bus a byte at a time at 2 Gbytes per second to an external memory. The timer is a 16-bit binary up-counter that counts NClck/2 (2 GHz) cycles. The external trace timer should be initialized to a value of 2^{16} - 1 - N, where N is the number of NClck/2 cycles between the 16-byte trace array reads. The maximum effective transfer rate in Gbytes/sec = 16 bytes × 2 Ghz / N. For example, x‘FFAF’ yields an interval of 80 2-GHz clock cycles, which would limit the peak external rate through the BIC or MIC to a peak internal data rate of 400 M bytes per second (16 bytes × 2 GHz / 80). A value of x‘FFEF’ yields the maximum rate of 2 Gbytes per second (16 bytes × 2 GHz / 16 cycles).
16:63	Reserved	Bits are not implemented; all bits read back zero.

9.3 Performance Monitor Only Registers

9.3.1 Performance Monitor Status/Interrupt Mask Register (pm_status)

This is a dual function register. When the register is written, a ‘1’ enables an associated interrupt, and a ‘0’ disables the associated interrupt. Reading this register clears the status bits and resets any pending associated performance-monitor interrupt. (Status read, mask write).

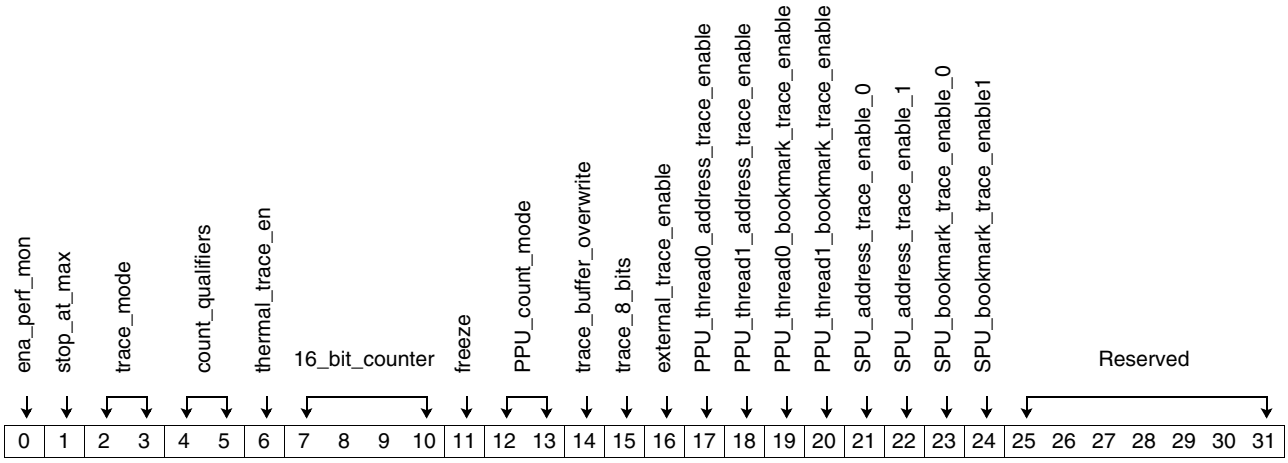
Register Short Name	pm_status	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	32 bits
Hex Offset From BE_MMIO_Base	x‘509400’	**Memory Map Area**	Pervasive: Performance Monitor
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



Bit(s)	Field Name	Description
0:7	counter_overflow	Counter [n] overflowed since the last read of this register. There is one bit for each of the eight performance monitor counters.
8	interval_time_out	The interval timer overflowed since the last read of this register.
9	trace_buffer_full	The trace buffer filled since the last read of this register.
10	trace_buffer_underflow	This bit is set when an MMIO read occurred from an empty local trace buffer since the last read of this register.
11:31	Reserved	Bits are not implemented; all bits read back zero

9.3.2 Performance Monitor Control Register (pm_control)

Register Short Name	pm_control	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	x‘509408’	**Memory Map Area**	Pervasive: Performance Monitor
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



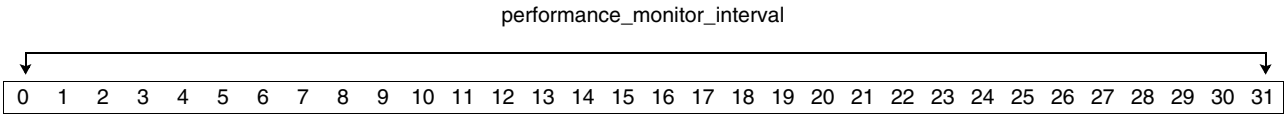
Bit(s)	Field Name	Description
0	ena_perf_mon	Enable performance monitor 0 Disable 1 Enable The performance monitor shares resources with the Trace Logic Analyzer, so software should enable only the Performance Monitor or the Trace Logic Analyzer, not both at the same time.
1	stop_at_max	Stop at maximum 0 Disable. Run the performance monitor counters through the maximum count (wrap) 1 Enable. Perform counter overflow occurrence tracing.
2:3	trace_mode	Trace mode. Pack performance-monitor data into 128 bits at each performance monitor interval timeout. Store the 128 bits to the trace buffer. For modes ‘01’ and ‘11’, must not read the pm_interval values through the MMIO. 00 No trace. Do not store performance-monitor data to the trace buffer. 01 Count trace. Store performance-monitor counter values to the trace buffer. Must set stop_at_max to ‘1’ and counter initial value to zero (x‘FF00’ for counting 8 bits of a 16-bit counter mode). 10 Occurrence trace. Store 64 bits of performance-monitor bus occurrence data to the trace buffer. 11 Threshold trace. Store 8 bits of performance monitor counter overflow data to the trace buffer. Must set stop_at_max to ‘1’. Counter freeze is not supported in this mode (bit[11] must be ‘0’).

Bit(s)	Field Name	Description
4:5	count_qualifiers	Count qualifiers 00 Do not use count qualifiers. 01 Allow the start of counting upon PM0 timeout; PM4 may not be used as a normal counter. 10 Allow the stop of counting upon PM4 timeout; PM0 may not be used as a normal counter. 11 Allow the start of counting upon PM0 timeout and stop upon PM4 timeout. When PM0 or PM4 are enabled as count qualifiers, they must be configured as two 16-bit counters. They stop at the maximum count, regardless of the state of the stop_at_max pm_control bit. PM1, PM2, PM3, PM5, PM6, and PM7 are subject to count qualification. After the stop count qualifier count reaches maximum count, all counters are held. The performance monitor must be disabled and count qualifiers reinitialized to use the count qualifiers again. If in count trace mode, the start and stop of count tracing is subject to the count qualifiers.
6	thermal_trace_en	Thermal trace enable 0 Disable 1 Enable. Enable thermal data to override data going to the trace array on bits[64:123]. The trace bits[0:63] may contain header and SPU address information per the trace data formats. 8-bit count tracing, occurrence tracing, and threshold tracing are not useful in conjunction with thermal tracing.
7:10	16_bit_counter	16-bit counter for pm0_4, pm1_5, pm2_6, pm3_7. For each bit: 0 Disable. Configure counter to act as a 32-bit counter 1 Enable. Configure counter to act as two 16-bit counters
11	freeze	Freeze all counters on overflow 0 Disable. Don't freeze. 1 Enable. Freeze all counters (including the interval timer) on any counter overflow, except the interval timer. When counters PM0 and PM4 are configured as count qualifiers, their overflowing does not cause a freeze. Freezing counts is not supported for the occurrence tracing and threshold tracing modes (see trace bits[2,3])
12:13	PPU_count_mode	PPU count mode. Count PPU performance-monitor signals only if the signal occurs while the PPU is in any of the following modes: 00 Supervisor mode 01 Hypervisor mode 10 Problem mode 11 Any of the above modes
14	trace_buffer_overwrite	Trace buffer overwrite 0 Disable. Don't overwrite. Performance monitor data is written until the trace buffer is full. When external trace is disabled, this provides a means to record performance data for the subsequent 1024 time intervals following a start event. Data may be read out of the trace buffer by setting the pm_control trace bits[2:3] to zero to prevent further writing to the buffer and then reading out the performance data from the trace buffer. When external trace is enabled, this allows for the use of the trace buffer as a FIFO for speed matching the trace buffer input and output data rates. 1 Enable. Overwrite trace buffer data when the trace buffer is full. This provides for continuous writing of counts to the buffer until the occurrence of a stop event. The data from the most recent 1024 count intervals can then be read out of the trace buffer. After overwriting, since the read pointer no longer points to the oldest data, the values of the write pointer can be used to determine the sequential order of the data. Do not set this bit if external trace is enabled.
15	trace_8_bits	Trace 8 bits of 16-bit counters 0 Disable 1 Enable. Trace only the lower 8 bits of the counters if the performance monitor is set for 16-bit count tracing. In this case, the counter must be initialized to x'FF00'. The 64-bit count data is written to the trace arrays (bits[64:127]) with the 8-bit count values in the following order: v0_v2_v4_v6_v1_v3_v5_v7.
16	external_trace_enable	Trace buffer output (enable external trace) 0 Disable. Count read from trace buffer via MMIO. 1 Enable. Performs external trace through BIC or MIC (MMIO reads from trace buffer not allowed when external trace enabled).

Bit(s)	Field Name	Description
17	PPU_thread0_address_trace_enable	PPU thread0 address trace enable 0 Disable 1 Enable. Enable PPU thread0 branch address for recent specific branch and link instructions to be stored to the trace buffer, in addition to the counter data stored at specified intervals. When the PPU address or a bookmark is enabled, then trace bits[0:15] are overwritten with header information.
18	PPU_thread1_address_trace_enable	PPU thread1 address trace enable 1 Enable. Enable PPU thread 1 branch address for recent specific branch and link instructions to be stored to the trace buffer in addition to the counter data stored at specified intervals. When the PPU address or a bookmark is enabled, then trace bits[0:15] are overwritten with header information.
19	PPU_thread0_bookmark_trace_enable	PPU thread0 bookmark trace enable 0 Disable 1 Enable. Enable PPU thread 0 bookmark SPR write data to be stored to the trace buffer, in addition to the counter data stored at specified intervals. When the PPU address or a bookmark is enabled, then trace bits[0:15] are overwritten with header information.
20	PPU_thread1_bookmark_trace_enable	PPU thread1 bookmark trace enable 0 Disable 1 Enable. Enable PPU thread 1 bookmark SPR write data to be stored to the trace buffer, in addition to the counter data stored at specified intervals. When the PPU address or a bookmark is enabled, then trace bits[0:15] are overwritten with header information.
21	SPU_address_trace_enable_0	SPU address trace enable 0 0 Disable 1 Enable. Overwrite trace bits[16:31] with the most recently received SPU program counter (address). Setup in the SPUx must be performed to route that particular SPU's address to the debug_bus event bit[1].
22	SPU_address_trace_enable_1	SPU address trace enable 1 0 Disable 1 Enable. Overwrite trace bits[32:47] with the most recently received SPU program counter (address). Setup in the SPUx must be performed to route that particular SPU's address to the debug_bus event bit[3].
23	SPU_bookmark_trace_enable_0	SPU bookmark trace enable 0 0 Disable 1 Enable. Store an SPU bookmark record to the trace buffer. Trace bits[0:15] are overwritten with header information and bits[16:31] with the most recently received SPU bookmark value. Setup in the SPUx must be performed to route that particular SPU's bookmark to the debug_bus event bit[1].
24	SPU_bookmark_trace_enable1	SPU bookmark trace enable1 0 Disable 1 Enable. Store an SPU bookmark record to the trace buffer. Trace bits[0:15] are overwritten with header information and bits[32:47] with the most recently received SPU bookmark value. Setup in the SPUx must be performed to route that particular SPU's bookmark to the debug_bus event bit[3].
25:31	Reserved	Bits are not implemented; all bits read back zero.

9.3.3 Performance Monitor Interval Register (pm_interval)

Register Short Name	pm_interval	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	32 bits
Hex Offset From BE_MMIO_Base	x‘509410’	**Memory Map Area**	Pervasive: Performance Monitor
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

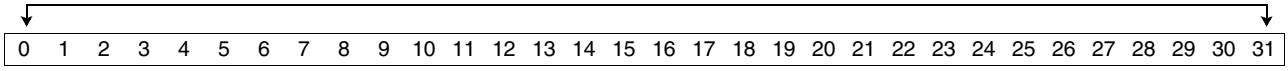


Bit(s)	Field Name	Description
0:31	performance_monitor_interval	Performance-monitor interval in core clock cycles (minimum interval is 10 cycles). This can be used in a trace mode to time the interval for writes to the trace buffer. This is a 32-bit binary up-counter. To program for an interval of N cycles, this counter must be initialized to a value of 2^{32} - N - 1. (For example, x‘FFFF FFF5’ gives an interval of 10 cycles). Write the initial value latch or read the current count through this MMIO address. **Note:** The total performance data rate, including potential address trace, must be lower than the trace buffer output rate (as determined by the external trace timer) to prevent the overflow of the trace buffer.

9.3.4 Performance Monitor Counter Pairs Registers (pmM_N)

Register Short Name	pm0_4 pm1_5 pm2_6 pm3_7	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	32 bits
Hex Offset From BE_MMIO_Base	x‘509418’ x‘509420’ x‘509428’ x‘509430’	**Memory Map Area**	Pervasive: Performance Monitor
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

performance_monitor_counter

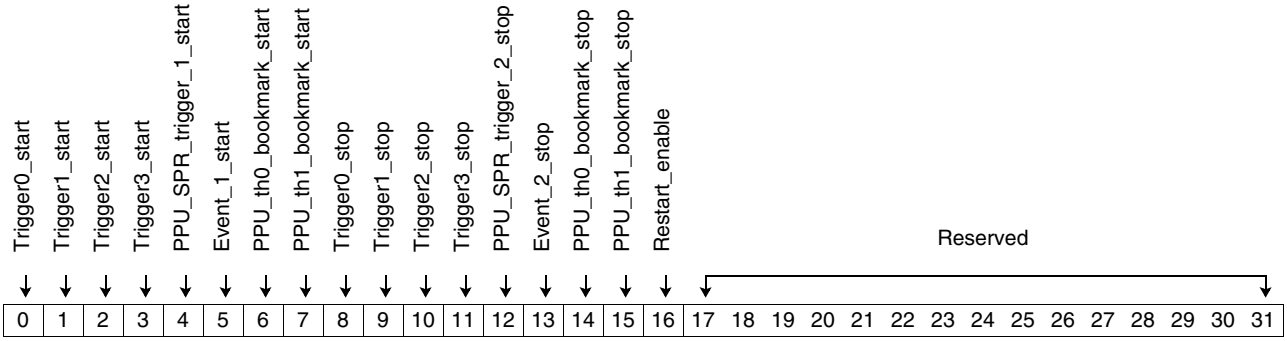


Bit(s)	Field Name	Description
0:31	performance_monitor_counter	32-bit count value or two 16-bit count values Performance-monitor counters 0, 1, 2, and 3 are read or written in either bits[0:15] or bits[0:31]. In 16-bit counter mode, performance-monitor counters 4, 5, 6, and 7 are read or written in bits[16:31]. In the count_trace or occurrence modes of counting, the performance-monitor counters must be written before the performance monitor is enabled, and values must not be read through MMIO. These are binary up-counters, so the value read indicates the number of events counted. For 8-bit occurrence counting, where N is the threshold value, the counter should be initialized to 2^{16} - 1 - N where N > 0. (for a 32-bit counter, this would be 2^{32} - 1 - N where N > 0). Write the initial value latch or read the current count through this MMIO address.

9.3.5 Performance Monitor Start Stop Register (pm_start_stop)

Start and stop for performance monitor counters. The start condition is an OR function of the start bits. The stop condition is an OR function of the stop bits. If the start qualifier is turned on, then the start signals act as a prequalifier to the start count qualifier. If the stop qualifier is turned on, then the stop signals act as a prequalifier to the stop count qualifier. The restart_enable signal allows a prequalifier start after a prequalifier stop.

Register Short Name	pm_start_stop	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	x'509438'	**Memory Map Area**	Pervasive: Performance Monitor
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



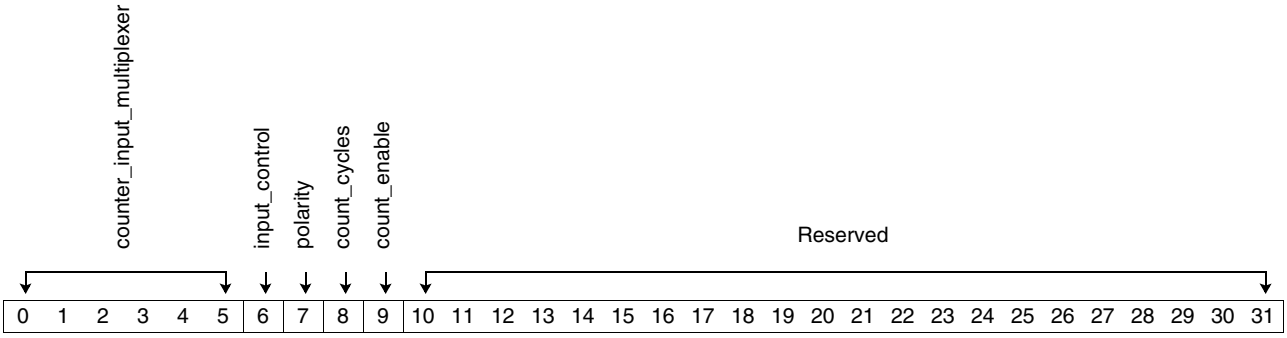
Bit(s)	Field Name	Description
0	Trigger0_start	Start counting upon debug bus trigger0.
1	Trigger1_start	Start counting upon debug bus trigger1.
2	Trigger2_start	Start counting upon debug bus trigger2.
3	Trigger3_start	Start counting upon debug bus trigger3.
4	PPU_SPR_trigger_1_start	Start counting upon PPU SPR trigger1.
5	Event_1_start	Start counting upon debug bus event 1.
6	PPU_th0_bookmark_start	Start counting upon PPU thread 0 (th0) bookmark start (req. bookmark enabled in pm_control).
7	PPU_th1_bookmark_start	Start counting upon PPU thread 1 (th1) bookmark start (req. bookmark enabled in pm_control).
8	Trigger0_stop	Stop counting upon debug bus trigger0.
9	Trigger1_stop	Stop counting upon debug bus trigger1.
10	Trigger2_stop	Stop counting upon debug bus trigger2.
11	Trigger3_stop	Stop counting upon debug bus trigger3.
12	PPU_SPR_trigger_2_stop	Stop counting upon PPU SPR trigger2.
13	Event_2_stop	Stop counting upon debug bus event2.

Bit(s)	Field Name	Description
14	PPU_th0_bookmark_stop	Stop counting upon PPU th0 bookmark stop (req. bookmark enabled in pm_control).
15	PPU_th1_bookmark_stop	Stop counting upon PPU th1 bookmark stop (req. bookmark enabled in pm_control).
16	Restart_enable	allows prequalifier start after a prequalifier stop; requires at least one start and one stop prequalifier to be set and the count qualifier (cq) start and cq stop (pm_control[4:5]) disabled.
17:31	Reserved	Bits are not implemented; all bits read back zero.

9.3.6 Performance Monitor Counter Control Registers (pmN_control)

These are the performance monitor counter control registers that allow control per counter. This is in contrast with the pm_control register, which applies across the entire performance monitor facility.

Register Short Name	pm0_control pm1_control pm2_control pm3_control pm4_control pm5_control pm6_control pm7_control	**Privilege Type**	Privilege 1
Access Type	MMIO Write Only	**Width**	32 bits
Hex Offset From BE_MMIO_Base	x'509440' x'509448' x'509450' x'509458' x'509460' x'509468' x'509470' x'509478'	**Memory Map Area**	Pervasive: Performance Monitor
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

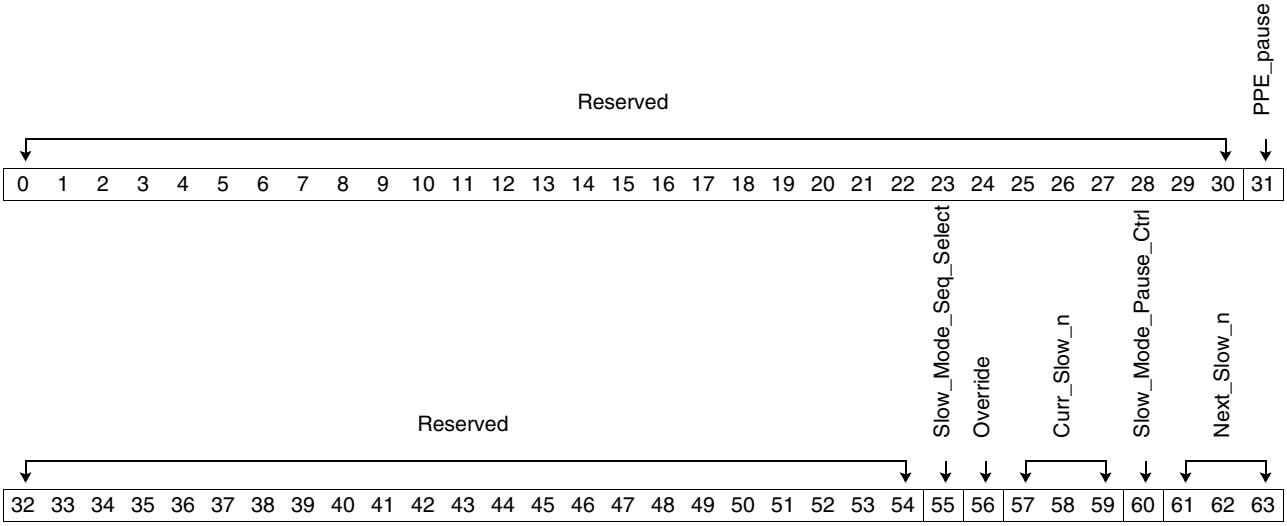


Bit(s)	Field Name	Description
0:5	counter_input_multiplexer	Counter Input Multiplexer 000000 Input is performance monitor bus bit 0 or trigger 0. 000001 Input is performance monitor bus bit 1 or trigger 1. 000010 Input is performance monitor bus bit 2 or trigger 2. 000011 Input is performance monitor bus bit 3 or trigger 3. 000100 Input is performance monitor bus bit 4 or event 0. 000101 Input is performance monitor bus bit 5 or event 1. 000110 Input is performance monitor bus bit 6 or event 2. 000111 Input is performance monitor bus bit 7 or event 3. 001000 Input is performance monitor bus bit 8 or external_trigger_in. 001001 Input is performance monitor bus bit 9 or spr_trigger1. 001010 Input is performance monitor bus bit 10 or spr_trigger2. 001011 Input is performance monitor bus bit 11. . . . The contents of the performance monitor bus are determined by the settings of the group_control and the debug_bus_control registers. In order to count cycles regardless of any input signal, event or trigger, set counter input mux (bits[0:5]) to '010000', input_control (bit[6]) to '1', polarity (bit[7]) =0. Also, see bit[6] (input_control).
6	input_control	Input control 0 Input is a performance-monitor bus bit selected by the counter input multiplexer (above). 1 Input is the alternative trigger or event selected by the counter input multiplexer (above).
7	polarity	Polarity 0 Negative polarity (count when 0, or, if counting edges, count negative transitions) 1 Positive polarity (count when 1, or, if counting edges, count positive transitions)
8	count_cycles	Count cycles 0 Count edges 1 Count cycles
9	count_enable	Count enable 0 Disable this counter 1 Enable this counter
10:31	Reserved	Bits are not implemented; all bits read back zero

9.4 Power Management Control Registers

9.4.1 Power Management Control Register (PMCR)

Register Short Name	PMCR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509880'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

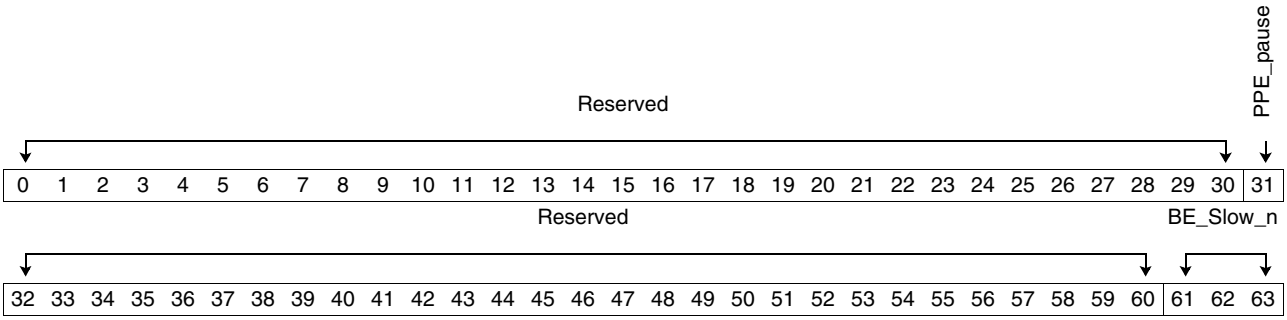


Bit(s)	Field Name	Description
0:30	Reserved	Bits are not implemented; all bits read back zero.
31	PPE_pause	Enables the PPE Pause (0) state. 0 Disable 1 Enable
32:54	Reserved	Bits are not implemented; all bits read back zero.
55	Slow_Mode_Seq_Select	Slow Mode Sequence Select 0 Transition frequencies in steps of 1/8 1 Transition frequencies in steps of 1/4
56	Override	Current Slow Mode Override 0 Slow mode 1 Overrides the current slow mode status stored in the hardware to the value in Curr_Slow(n) bits[57:59]. Hardware resets this bit to '0' after the update completes. **Note:** Setting this bit might indirectly cause a slow mode transition if the Curr_Slow(n) setting is different than the Next_Slow setting. If so, hardware first updates the Slow_Mode status register to the Curr_Slow(n) value, and then transitions to the Next_Slow bit value.

Bit(s)	Field Name	Description
57:59	Curr_Slow_n	Current CBE Slow (n) State Controls the current CBE Slow (n) state. These bits encode the CBE core frequencies. **Value** **n** **CBE Core Frequency** (NClk) 000 0 Max 001 1 Max/2 010 2 Max/3 011 3 Max/4 100 4 Max/5 101 5 Max/6 110 6 Max/8 111 7 Max/10 **Note:** These bit settings cannot change value until the corresponding bits in the PMSR register match these bit settings.
60	Slow_Mode_Pause_Ctrl	Slow Mode Pause Control 0 System switches to new slow mode setting without waiting for pause mode. 1 System waits for pause mode before switching to new slow mode setting.
61:63	Next_Slow_n	Next CBE Slow (n) State Controls the next Slow (n) state at the CBE level. This field encodes the CBE core frequencies. **Value** **n** **CBE Core Frequency** (NClk) 000 0 Max 001 1 Max/2 010 2 Max/3 011 3 Max/4 100 4 Max/5 101 5 Max/6 110 6 Max/8 111 7 Max/10 **Note:** These bit settings cannot change value until the corresponding bits in the PMSR register match these bit settings.

9.4.2 Power Management Status Register (PMSR)

Register Short Name	PMSR	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509888'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



Bit(s)	Field Name	Description
0:30	Reserved	Bits are not implemented; all bits read back zero.
31	PPE_pause	Enables the PPE Pause (0) state.
32:60	Reserved	Bits are not implemented; all bits read back zero.
61:63	BE_Slow_n	CBE Slow (n) State Status Status for the CBE Slow (n) state. **Value** **n** **CBE Core Frequency** (NClk) 000 0 Max 001 1 Max/2 010 2 Max/3 011 3 Max/4 100 4 Max/5 101 5 Max/6 110 6 Max/8 111 7 Max/10

9.5 Thermal Management MMIO Registers

Thermal Monitor Control (TMC) logic is capable of generating a thermal interrupt. The thermal interrupt is defined in the *PowerPC Architecture* and has its dedicated interrupt vector.

The thermal sensor interrupt registers control the generation of a thermal management interrupt to the PPE. This set of registers consist of the thermal sensor interrupt temperature registers (TS_ITR1 and TS_ITR2), the thermal sensor interrupt status register (TS_ISR), and the thermal sensor interrupt mask register (TS_IMR).

Table 9-2. Encode to Temperature Mapping

Thermal Sensor Temperature Encoding			
6-bit Encode	Temperature Range	6-bit Encode	Temperature Range
0	Temp ≤ 65° C	16	95° C ≤ Temp < 97° C
1	65° C ≤ Temp < 67° C	17	97° C ≤ Temp < 99° C
2	67° C ≤ Temp < 69° C	18	99° C ≤ Temp < 101° C
3	69° C ≤ Temp < 71° C	19	101° C ≤ Temp < 103° C
4	71° C ≤ Temp < 73° C	20	103° C ≤ Temp < 105° C
5	73° C ≤ Temp < 75° C	21	105° C ≤ Temp < 107° C
6	75° C ≤ Temp < 77° C	22	107° C ≤ Temp < 109° C
7	77° C ≤ Temp < 79° C	23	109° C ≤ Temp < 111° C
8	79° C ≤ Temp < 81° C	24	111° C ≤ Temp < 113° C
9	81° C ≤ Temp < 83° C	25	113° C ≤ Temp < 115° C
10	83° C ≤ Temp < 85° C	26	115° C ≤ Temp < 117° C
11	85° C ≤ Temp < 87° C	27	117° C ≤ Temp < 119° C
12	87° C ≤ Temp < 89° C	28	119° C ≤ Temp < 121° C
13	89° C ≤ Temp < 91° C	29	121° C ≤ Temp < 123° C
14	91° C ≤ Temp < 93° C	30	123° C ≤ Temp < 125° C
15	93° C ≤ Temp < 95° C	31 – 63	125° C ≤ Temp

Note: Refer to the *Cell Broadband Engine Datasheet* for the accuracy of the Digital Thermal Sensor.

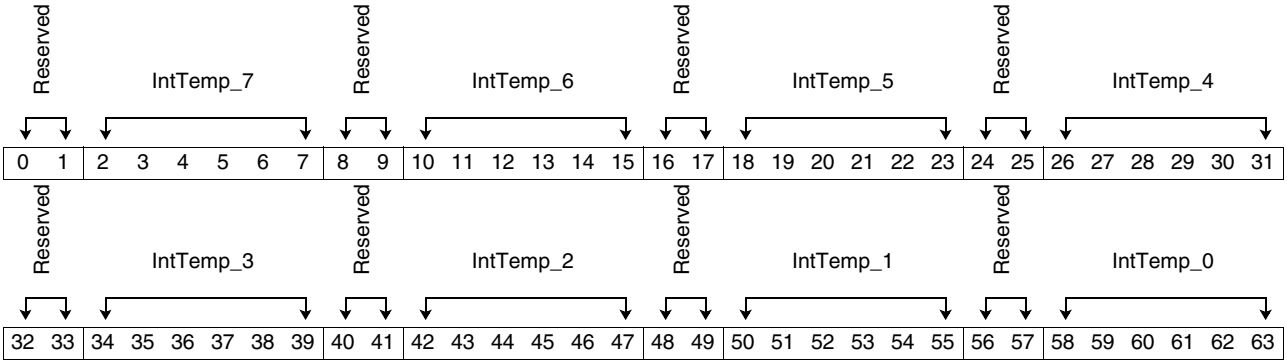
9.5.1 Thermal Sensor Interrupt Temperature Register 1 (TS_ITR1)

The thermal sensor interrupt temperature register one (TS_ITR1) contains the interrupt temperature level for the sensors located in the SPEs. The encoded interrupt temperature levels in this register are compared to the corresponding interrupt temperature encoding in TS_CTSR1. The result of these comparisons are used to generate a thermal management interrupt. Each sensor's interrupt temperature level is independent.

Note: See *Table 9-2* on page 138 for mappings of the temperature encodings for the following bit ranges/field names:

bits[2:7]	IntTemp(7)	bits[34:39]	IntTemp(3)
bits[10:15]	IntTemp(6)	bits[42:47]	IntTemp(2)
bits[18:23]	IntTemp(5)	bits[50:55]	IntTemp(1)
bits[26:31]	IntTemp(4)	bits[58:63]	IntTemp(0)

Register Short Name	TS_ITR1	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509820'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



Bit(s)	Field Name	Description
0:1	Reserved	Bits are not implemented; all bits read back zero.
2:7	IntTemp_7	Temperature level at which digital thermal sensor 7 causes an interrupt. Digital thermal sensor 7 is located in physical SPE 7.
8:9	Reserved	Bits are not implemented; all bits read back zero.
10:15	IntTemp_6	Temperature level at which digital thermal sensor 6 causes an interrupt. Digital thermal sensor 6 is located in physical SPE 6.
16:17	Reserved	Bits are not implemented; all bits read back zero.
18:23	IntTemp_5	Temperature level at which digital thermal sensor 5 causes an interrupt. Digital thermal sensor 5 is located in physical SPE 5.
24:25	Reserved	Bits are not implemented; all bits read back zero.
26:31	IntTemp_4	Temperature level at which digital thermal sensor 4 causes an interrupt. Digital thermal sensor 4 is located in physical SPE 4.

Bit(s)	Field Name	Description
32:33	Reserved	Bits are not implemented; all bits read back zero.
34:39	IntTemp_3	Temperature level at which digital thermal sensor 3 causes an interrupt. Digital thermal sensor 3 is located in physical SPE 3.
40:41	Reserved	Bits are not implemented; all bits read back zero.
42:47	IntTemp_2	Temperature level at which digital thermal sensor 2 causes an interrupt. Digital thermal sensor 2 is located in physical SPE 2.
48:49	Reserved	Bits are not implemented; all bits read back zero.
50:55	IntTemp_1	Temperature level at which digital thermal sensor 1 causes an interrupt. Digital thermal sensor 1 is located in physical SPE 1.
56:57	Reserved	Bits are not implemented; all bits read back zero.
58:63	IntTemp_0	Temperature level at which digital thermal sensor 0 causes an interrupt. Digital thermal sensor 0 is located in physical SPE 0.

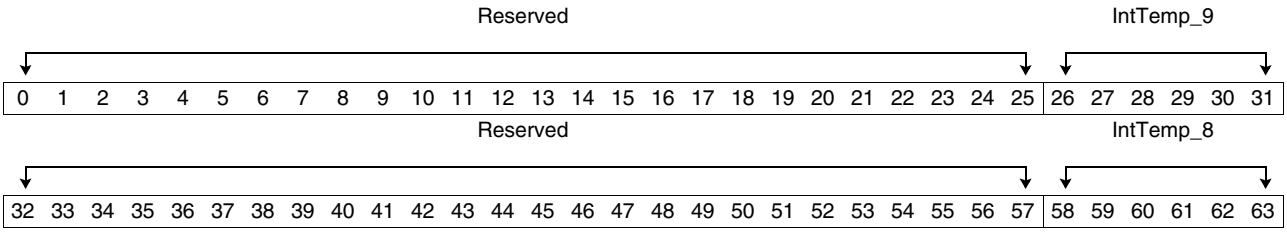
9.5.2 Thermal Sensor Interrupt Temperature Register 2 (TS_ITR2)

This register contains the interrupt temperature level for the sensors located in the PPE and adjacent to the linear thermal sensor. The encoded interrupt temperature levels in this register are compared to the corresponding interrupt temperature encoding in TS_CTSR2. The result of these comparisons are used to generate a thermal management interrupt. Each sensor's interrupt temperature level is independent.

Note: See *Table 9-2* on page 138 for mappings of the temperature encodings for the following bit ranges and field names:

bits[26:31] IntTemp(9)
bits[58:63] IntTemp(8)

Register Short Name	TS_ITR2	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509828'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



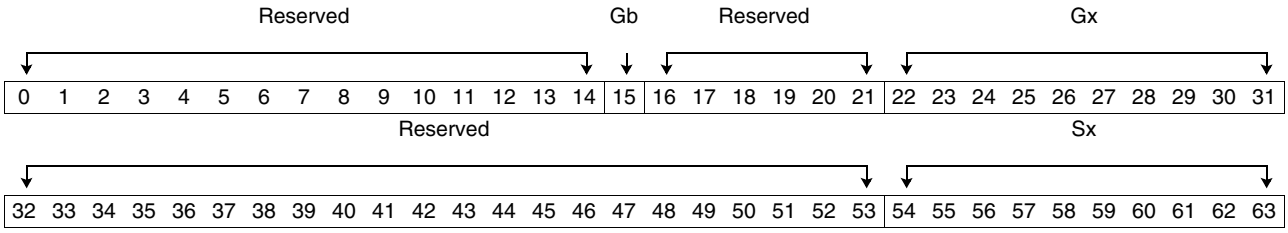
Bit(s)	Field Name	Description
0:25	Reserved	Bits are not implemented; all bits read back zero.
26:31	IntTemp_9	Temperature level at which digital thermal sensor 9 causes an interrupt. Digital thermal sensor 9 is located adjacent to the linear thermal sensor.
32:57	Reserved	Bits are not implemented; all bits read back zero.
58:63	IntTemp_8	Temperature level at which digital thermal sensor 8 causes an interrupt. Digital thermal sensor 8 is located in the PPU.

9.5.3 Thermal Sensor Interrupt Status Register (TS_ISR)

The thermal sensor interrupt status register (TS_ISR) identifies which sensors met the interrupt condition. This register contains two set of status bits; the digital sensor global threshold interrupt status bits (TS_ISR[22:31]) and the digital sensor threshold interrupt status (TS_ISR[54:63]).

Hardware sets the status bits as outlined below. Once a status bit is set to '1', the state is maintained until reset to '0' by privileged software. Privileged software resets a status bit to '0' by writing a '1' to the corresponding bit in the TS_ISR.

Register Short Name	TS_ISR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509838'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



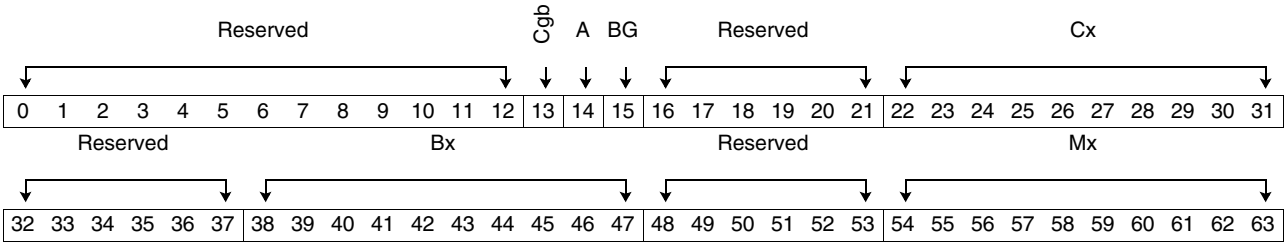
Bit(s)	Field Name	Description
0:14	Reserved	Bits are not implemented. All bits read back zero.
15	Gb	Digital Sensor Global Below Threshold interrupt status. This bit is not set when the interrupt range is configured to above or equal to the global interrupt temperature level. 0 Interrupt not pending for Digital Thermal Sensor. 1 Interrupt pending for Digital Thermal Sensor below the global threshold interrupt level.
16:21	Reserved	Bits are not implemented. All bits read back zero.
22:31	Gx	Digital Sensor Global Threshold interrupt status (where x equals the sensor number). These bits are not set when the interrupt range is configured to below the global interrupt temperature level. The sensor numbers range left to right from 9 to 0, so that bit 22 is associated with sensor number 9 and bit 31 is associated with sensor number 0. See TS_CTSR1 and TS_CTSR2 for sensor number mapping. For each bit: x'000' Interrupt not pending for Digital Thermal Sensor x. x'001' Interrupt pending for Digital Thermal Sensor x.
32:53	Reserved	Bits are not implemented. All bits read back zero.
54:63	Sx	Digital Sensor Threshold interrupt status (where x equals sensor number). The sensor numbers range left to right from 9 to 0, so that bit 54 is associated with sensor number 9 and bit 63 is associated with sensor number 0. See TS_CTSR1 and TS_CTSR2 for sensor number mapping. For each bit: x'000' Interrupt not pending for Digital Thermal Sensor x. x'001' Interrupt pending for Digital Thermal Sensor x.

Programming Note: If the interrupt condition is still met for a status bit being reset, the status bit remains set. To avoid an immediate interrupt, privileged software should either mask the interrupting condition using the TS_IMR register or ensure that the interrupting condition is not met before resetting the interrupt status bit.

9.5.4 Thermal Sensor Interrupt Mask Register (TS_IMR)

The thermal sensor interrupt mask register contains mask bits (M_x) to prevent an interrupt status bit from generating a thermal management interrupt to the PPE It also contains controls (Bx) to select the temperature range for the interrupt conditions. In addition, this register contains controls for (Cx) to select which sensors participate in the global interrupt condition.

Register Short Name	TS_IMR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509840'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV



Bit(s)	Field Name	Description
0:12	Reserved	Bits are not implemented; all bits read back zero.
13	Cgb	Mask for Digital Thermal Global Below Threshold interrupt 0 Sensor will not cause a global interrupt (disabled). 1 Sensor may cause a global interrupt (enabled). **Note:** This bit only affects the interrupt generation when the global interrupt is set for interrupting when all sensors are below the threshold.
14	A	Enable for assertion of system controller "Attention" signal 0 Attention will not be asserted (disabled). 1 Attention will be asserted when the Gb or any Gx bit is set (enabled).
15	BG	Direction for Digital Thermal Global Threshold interrupt 0 Interrupt when temperature of any enabled sensor is above or equal to the global Interrupt temperature level. 1 Interrupt when temperatures of all enabled sensors are below the global interrupt temperature level.
16:21	Reserved	Bits are not implemented; all bits read back zero.

Bit(s)	Field Name	Description
22:31	Cx	Mask for Digital Thermal Global Threshold interrupt (where x equals sensor number). The sensor numbers range left to right from 9 to 0, so that bit 22 is associated with sensor number 9 and bit 31 is associated with sensor number 0. See TS_CTSR1 and TS_CTSR2 for sensor number mapping. For each bit: 0 Sensor will not cause a global interrupt (disabled). 1 Sensor may cause a global interrupt (enabled).
32:37	Reserved	Bits are not implemented; all bits read back zero.
38:47	Bx	Direction for Digital Thermal Threshold interrupt (where x equals sensor number). The sensor numbers range left to right from 9 to 0, so that bit 38 is associated with sensor number 9 and bit 47 is associated with sensor number 0. See TS_CTSR1 and TS_CTSR2 for sensor number mapping. For each bit: 0 Interrupt when temperature is above or equal to the interrupt temperature level. 1 Interrupt when temperature is below the interrupt temperature level.
48:53	Reserved	Bits are not implemented; all bits read back zero.
54:63	Mx	Mask for Digital Thermal Threshold interrupt (where x equals sensor number). The sensor numbers range left to right from 9 to 0, so that bit 54 is associated with sensor number 9 and bit 63 is associated with sensor number 0. See TS_CTSR1 and TS_CTSR2 for sensor number mapping. For each bit: 0 Interrupt is disabled. 1 Interrupt is enabled.

9.5.5 Thermal Management System Interrupt Mask Register (TM_SIMR)

This register controls which PPE interrupts cause the thermal management logic to exit a throttling state on the PPE. Throttling is exited for both PPE threads, regardless of the thread targeted by the interrupt. Throttling of the SPEs is never exited based on a system interrupt condition. The PPE interrupt conditions that can override a throttling condition are listed below:

- External
- Decrementer
- Hypervisor Decrementer
- System Error
- Thermal Management

Register Short Name	TM_SIMR	**Privilege Type**	Privilege 1
Access Type	MMIO Read/Write	**Width**	64 bits
Hex Offset From BE_MMIO_Base	x'509858'	**Memory Map Area**	Pervasive: Thermal and Power Management
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	PRV

Bit(s)	Field Name	Description
0:58	Reserved	Bits are not implemented; all bits read back zero.
59	T	Mask for Thermal Management Interrupt 0 Interrupt does not exit PPE throttling state (disabled). 1 Interrupt exits the PPE throttling state (enabled).
60	S	Mask for System Error Interrupt 0 Interrupt does not exit PPE throttling state (disabled). 1 Interrupt exits the PPE throttling state (enabled).
61	H	Mask for Hypervisor Decrementer Interrupt 0 Interrupt does not exit PPE throttling state (disabled). 1 Interrupt exits the PPE throttling state (enabled).
62	D	Mask for Decrementer Interrupt 0 Interrupt does not exit PPE throttling state (disabled). 1 Interrupt exits the PPE throttling state (enabled).
63	E	Mask for External Interrupt 0 Interrupt does not exit PPE throttling state (disabled). 1 Interrupt exits the PPE throttling state (enabled).

10. PPE Special Purpose Registers

This section describes Special Purpose Registers (SPRs) used by the PowerPC Processor Element (PPE). These registers are read from or written to using the **mfspr** and **mtspr** PowerPC instructions, respectively. *Table 10-1* on page 148 shows the PPE SPR memory map and lists the PPE SPRs.

Table 10-1 uses the following conventions:

- Architected SPRs that do not contain implementation-specific information are included in *Table 10-1*, but are not described in this manual. *Table 10-1* provides cross references to additional information for each architected SPR.
- Architected SPRs that contain implementation-specific information are included in *Table 10-1* and are described in this section. When applicable, specific cross references to additional information are provided in each register description. The SPRs in the *Decimal* column are highlighted to indicate that an SPR is implementation specific.

  Programming Note: Programmers should be aware that code using implementation-specific information is not guaranteed to be portable across different implementations of the architecture, although a significant effort is made to minimize incompatibilities between different designs.
- The notes provided at the end of *Table 10-1* describe unique and implementation-specific information for certain SPRs.

Table 10-1. PPE Special Purpose Registers (Page 1 of 5)

SPR		Duplicated for Multithreading[3]	Register Name (Short Name) • *Cross Reference to Additional Information*	Unit	Read/Write	Synchronization Requirements[4]				Hypervisor/ Privileged[5]		Size (bits)	Power-on Reset (POR) Value (All bits set to '0' unless otherwise noted)
						For Data		For Instructions					
Decimal[1]	spr[5:9] spr[0:4][2]					Before Writes	After Writes	Before Writes	After Writes	Read (mf)	Write (mt)		
01	00000 00001	Yes	*Fixed-Point Exception Register (XER)* • *PowerPC User Instruction Set Architecture, Book I*	XU	R/W	N/A				—	—	64	
08	00000 01000	Yes	*Link Register (LR)* • *PowerPC User Instruction Set Architecture, Book I*	IU	R/W	N/A				—	—	64	
09	00000 01001	Yes	*Count Register (CTR)* • *PowerPC User Instruction Set Architecture, Book I*	IU	R/W	N/A				—	—	64	
18	00000 10010	Yes	*Data Storage Interrupt Status Register (DSISR)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				Priv		32	
19	00000 10011	Yes	*Data Address Register (DAR)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				Priv		64	
22	00000 10110	Yes	*Decrementer Register (DEC)* • *PowerPC Operating Environment Architecture, Book III*	MMU	R/W	None				Priv		32	x'7FFF_FFFF'
25	00000 11001	No	*Storage Description Register 1 (SDR1)* • *PowerPC Operating Environment Architecture, Book III*	MMU	R/W	pte-sync	CSI	pte-sync	CSI	HV		64	
26	00000 11010	Yes	*Machine Status Save/Restore Register 0 (SRR0)* • *PowerPC Operating Environment Architecture, Book III*	IU	R/W	N/A				Priv		64	
27	00000 11011	Yes	*Machine Status Save/Restore Register 1 (SRR1)* • *PowerPC Operating Environment Architecture, Book III*	IU	R/W	N/A				Priv		64	
29	00000 11101	Yes	*Address Compare Control Register (ACCR)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	CSI		None		Priv		64	
136	00100 01000	No	*Control Register (CTRL)* • *Section 10.1.1 Control Register (CTRL) on page 153*	IU	R	N/A				—	N/A	32	
152	00100 11000				W	None				N/A	Priv[6]		N/A
256	01000 00000	Yes	*VXU Register Save (VRSAVE)* • *See the PowerPC Microprocessor Family: Vector/SIMD Multimedia Extension Technology Programming Environments Manual*	XU	R/W	N/A				—	—	32	
259	01000 00011	Yes	*Software Use Special Purpose Register 3 (SPRG3) - Read Only* • *PowerPC Operating Environment Architecture, Book III*	XU	R	N/A				—	N/A	64	
268	01000 01100	No	*Time Base Register (TB)* • *PowerPC Operating Environment Architecture, Book III* The physical implementation of this register includes the following registers: • *Time Base Register - Read Only (TB), SPR #268* • *Time Base Upper Register - Read Only (TBU), SPR #269* • *Time Base Lower Register - Write Only (TBL), SPR #284* • *Time Base Upper Register - Write Only (TBU), SPR #285*	MMU	R	N/A				—	N/A	64	
269	01000 01101											32	

Table 10-1. PPE Special Purpose Registers (Page 2 of 5)

SPR		Duplicated for Multithreading[3]	Register Name (Short Name) • *Cross Reference to Additional Information*	Unit	Read/Write	Synchronization Requirements[4]				Hypervisor/ Privileged[5]		Size (bits)	Power-on Reset (POR) Value (All bits set to '0' unless otherwise noted)
						For Data		For Instructions					
Decimal[1]	spr[5:9] spr[0:4][2]					Before Writes	After Writes	Before Writes	After Writes	Read (mf)	Write (mt)		
272	01000 10000	Yes	*Software Use Special Purpose Register 0 (SPRG0)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				Priv		64	
273	01000 10001	Yes	*Software Use Special Purpose Register 1 (SPRG1)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				Priv		64	
274	01000 10010	Yes	*Software Use Special Purpose Register 2 (SPRG2)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				Priv		64	
275	01000 10011	Yes	*Software Use Special Purpose Register 3 (SPRG3)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				Priv		64	
284	01000 11100	No	*Time Base Register (TB)* • *PowerPC Operating Environment Architecture, Book III* The physical implementation of this register includes the following registers: • *Time Base Register - Read Only (TB), SPR #268* • *Time Base Upper Register - Read Only (TBU), SPR #269* • *Time Base Lower Register - Write Only (TBL), SPR #284* • *Time Base Upper Register - Write Only (TBU), SPR #285*	MMU	W	None				N/A	HV	32	N/A
285	01000 11101											32	
287	01000 11111	No	*PPE Processor Version Register (PVR)* • *PowerPC Operating Environment Architecture, Book III*	XU	R	N/A				Priv	N/A	32	x'0070_0400' (DD2) x'0070_0100' (DD1)
304	01001 10000	Yes	*Hypervisor Software Use Special Purpose Register 0 (HSPRG0)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				HV		64	
305	01001 10001	Yes	*Hypervisor Software Use Special Purpose Register 1 (HSPRG1)*[7] • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	N/A				HV		64	
310	01001 10110	No	*Hypervisor Decrementer Register (HDEC)*[7] • *PowerPC Operating Environment Architecture, Book III*	MMU	R/W	None				HV		32	x'7FFF_FFFF'
312	01001 11000	No	*Real Mode Offset Register (RMOR)* • *Section 10.1.2 Real Mode Offset Register (RMOR) on page 155*	MMU	R/W	CSI		None	CSI	HV		64	
313	01001 11001	No	*Hypervisor Real Mode Offset Register (HRMOR)*[7] • *Section 10.1.3 Hypervisor Real Mode Offset Register (HRMOR) on page 156*	MMU	R/W	CSI		None	CSI	HV		64	
314	01001 11010	Yes	*Hypervisor Machine Status Save/Restore Register 0 (HSRR0)*[7] • *PowerPC Operating Environment Architecture, Book III*	IU	R/W	N/A				HV		64	
315	01001 11011	Yes	*Hypervisor Machine Status Save/Restore Register 1 (HSRR1)*[7] • *PowerPC Operating Environment Architecture, Book III*	IU	R/W	N/A				HV		64	
318	01001 11110	Par-tial	*Logical Partition Control Register (LPCR)*[7] • *Section 10.1.4 Logical Partition Control Register (LPCR) on page 157*	MMU	R/W	CSI		None	CSI	HV		64	

Table 10-1. PPE Special Purpose Registers (Page 3 of 5)

SPR		Duplicated for Multithreading[3]	Register Name (Short Name) • *Cross Reference to Additional Information*	Unit	Read/Write	Synchronization Requirements[4]				Hypervisor/ Privileged[5]		Size (bits)	Power-on Reset (POR) Value (All bits set to '0' unless otherwise noted)
						For Data		For Instructions					
Decimal[1]	spr[5:9] spr[0:4][2]					Before Writes	After Writes	Before Writes	After Writes	Read (mf)	Write (mt)		
319	01001 11111	No	*Logical Partition Identity Register (LPIDR)*[7] • *Section 10.1.5 Logical Partition Identity Register (LPIDR) on page 158*	MMU	R/W	CSI		CSI		HV		32	
896	11100 00000	Yes	*Thread Status Register Local (TSRL)* • *Section 10.1.6 Thread Status Register Local (TSRL) on page 159*	IU	R/W	None			CSI	—	—	64	
897	11100 00001	Yes	*Thread Status Register Remote (TSRR)* • *Section 10.1.7 Thread Status Register Remote (TSRR) on page 161*	IU	R	N/A				—	N/A	64	
921	11100 11001	No	*Thread Switch Control Register (TSCR)* • *Section 10.1.8 Thread Switch Control Register (TSCR) on page 162*	IU	R/W	None			CSI	HV		64	
922	11100 11010	No	*Thread Switch Timeout Register (TTR)* • *Section 10.1.9 Thread Switch Timeout Register (TTR) on page 164*	IU	R/W	None			CSI	HV		64	
946	11101 10010	Yes	*PPE Translation Lookaside Buffer Index Hint Register (PPE_TLB_Index_Hint)* • *Section 10.1.10.1 PPE Translation Lookaside Buffer Index Hint Register (PPE_TLB_Index_Hint) on page 165*	MMU	R	N/A				Priv	N/A	64	
947	11101 10011	No	*PPE Translation Lookaside Buffer Index Register (PPE_TLB_Index)* • *Section 10.1.10.2 PPE Translation Lookaside Buffer Index Register (PPE_TLB_Index) on page 167*	MMU	R/W[8]	None				HV		64	
948	11101 10100	No	*PPE Translation Lookaside Buffer Virtual-Page Number Register (PPE_TLB_VPN)* • *Section 10.1.10.3 PPE Translation Lookaside Buffer Virtual-Page Number Register (PPE_TLB_VPN) on page 169*	MMU	R/W	CSI	CSI	None	CSI	HV		64	
949	11101 10101	No	*PPE Translation Lookaside Buffer Real-Page Number Register (PPE_TLB_RPN)* • *Section 10.1.10.4 PPE Translation Lookaside Buffer Real-Page Number Register (PPE_TLB_RPN) on page 170*	MMU	R/W	None				HV		64	
951	11101 10111	No	*PPE Translation Lookaside Buffer RMT Register (PPE_TLB_RMT)* • *Section 10.1.11 PPE Translation Lookaside Buffer RMT Register (PPE_TLB_RMT) on page 172*	MMU	R/W	CSI	CSI	None	CSI	HV		64	
952	11101 11000	N/A	*Reserved*										
953	11101 11001	N/A	*Reserved*										
954	11101 11010	Yes	*Data Class ID Register 0 (DCIDR0)* • *Section 10.1.12.1 Data Class ID Register 0 (DCIDR0) on page 173* • *Cell Broadband Engine Architecture*	XU	R/W	Sync	Sync, CSI[9]	None	None	HV		32	
955	11101 11011	N/A	*Reserved*										
956	11101 11100	N/A	*Reserved*										
957	11101 11101	Yes	*Data Class ID Register 1 (DCIDR1)* • *Section 10.1.12.2 Data Class ID Register 1 (DCIDR1) on page 173*	XU	R/W	Sync	Sync, CSI[9]	None	None	HV		32	

Table 10-1. PPE Special Purpose Registers (Page 4 of 5)

SPR		Duplicated for Multithreading[3]	Register Name (Short Name) • *Cross Reference to Additional Information*	Unit	Read/Write	Synchronization Requirements[4]				Hypervisor/ Privileged[5]		Size (bits)	Power-on Reset (POR) Value (All bits set to '0' unless otherwise noted)
						For Data		For Instructions					
Decimal[1]	spr[5:9] spr[0:4][2]					Before Writes	After Writes	Before Writes	After Writes	Read (mf)	Write (mt)		
976	11110 10000	N/A	*Reserved*										
977	11110 10001	N/A	*Reserved*										
978	11110 10010	Yes	*Instruction Class ID Register 0 (ICIDR0)* • *Section 10.1.13.1 Instruction Class ID Register 0 (ICIDR0) on page 174*	IU	R/W	None	None	Sync	Sync, CSI[9]	HV		32	
979	11110 10011	N/A	*Reserved*										
980	11110 10100	N/A	*Reserved*										
981	11110 10101	Yes	*Instruction Class ID Register 1 (ICIDR1)* • *Section 10.1.13.2 Instruction Class ID Register 1 (ICIDR1) on page 175*	IU	R/W	None	None	Sync	Sync, CSI[9]	HV		32	
1008	11111 10000	N/A	Reserved										
1009	11111 10001	N/A	Reserved										
1012	11111 10100												
1013	11111 10101	Yes	*Data Address Breakpoint Register (DABR)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	Sync	CSI	None		HV		64	
1014	11111 10110	N/A	Reserved										
1015	11111 10111	Yes	*Data Address Breakpoint Register Extension (DABRX)* • *PowerPC Operating Environment Architecture, Book III*	XU	R/W	Sync	CSI	None		HV		64	
1016	11111 11000	N/A	Reserved										
1017	11111 11001												
1018	11111 11010												
1019	11111 11011												
1020	11111 11100												
1021	11111 11101												
1022	11111 11110	No	*CBEA-Compliant Processor Version Register (BP_VR)* • *Section 10.1.14 CBEA-Compliant Processor Version Register (BP_VR) on page 176* • *Cell Broadband Engine Architecture*	XU	R	N/A				Priv	N/A	64	
1023	11111 11111	Yes	*Processor Identification Register (PIR)*[7] • *Section 10.1.15 Processor Identification Register (PIR) on page 177*	XU	R	N/A				Priv	N/A	32	See *Section 10.1.15*

Table 10-1. PPE Special Purpose Registers (Page 5 of 5)

SPR		Duplicated for Multithreading[3]	Register Name (Short Name) • *Cross Reference to Additional Information*	Unit	Read/Write	Synchronization Requirements[4]				Hypervisor/ Privileged[5]		Size (bits)	Power-on Reset (POR) Value (All bits set to '0' unless otherwise noted)
						For Data		For Instructions					
Decimal[1]	spr[5:9] spr[0:4][2]					Before Writes	After Writes	Before Writes	After Writes	Read (mf)	Write (mt)		
N/A	N/A	Yes	*Machine State Register (MSR)* • *Section 10.2.1 Machine State Register (MSR) on page 178*	IU	R/W	N/A				Priv	—		

1. Implementation-specific SPRs are highlighted.
2. The order of the two 5-bit halves of the SPR number is reversed (to follow the convention of the architecture documents).
3. Any register that is nonduplicated per thread requires special care by the hypervisor when written. The hypervisor must not cause an implicit branch or undefined behavior for the thread that is not writing the register.
4. For more information, see the *Synchronization Requirements for Context Alterations* section of the *PowerPC Operating Environment Architecture, Book III.* In this column, Sync refers to the lightweight **sync L=1** instruction unless otherwise indicated.
5. Explanation of the *Hypervisor/Privileged* column:
   - HV: Indicates that the register is a hypervisor resource. The hypervisor state must be enabled (MSR[HV] = '1') to write this register. Attempts to modify the contents of a hypervisor resource (such as using the move-to SPR instruction) in privileged, but nonhypervisor state (MSR[HV, PR] = '00') cause a privileged-instruction program interrupt.
   - Priv: Indicates that the register is privileged. The privileged state must be enabled (MSR[PR] = '0') to read or write to the register. Attempts to access the contents of a privileged resource (such as using the move-to SPR or move-from SPR instruction) in nonprivileged state (MSR[PR] = '1') cause a privileged-instruction program interrupt.
   - —: Indicates that the register is neither privileged nor a hypervisor resource.
   - N/A: Indicates that the register is either read-only or write-only.
     – Writes using move-to instructions to unimplemented SPRs are treated as **nop** instructions. Architected registers are not changed.
     – Writes using move-to instructions to read-only SPRs are treated like unimplemented SPRs.
     – Reads using move-from instructions from unimplemented SPRs cause zeros to be written back to the General Purpose Register (GPR).
     – Reads using move-from instructions from write-only SPRs are treated like unimplemented SPRs.
6. The Thread Enable Bits field of CTRL can be modified only in hypervisor mode. Attempts to modify (using the move-to SPR instruction) the Thread Enable Bits field of the CTRL register while not in hypervisor mode (MSR[HV] = '0') are ignored.
7. These registers are for logical partitioning (LPAR) support.
8. Reading of these registers is allowed for diagnostic purposes.
9. Indicates a **sync** instruction followed by any context synchronizing instruction.

10.1 SPR Definitions

This section describes the implementation-specific special purpose registers (SPRs) used in this implementation. For a complete listing of SPRs, see *Table 10-1 PPE Special Purpose Registers* on page 148.

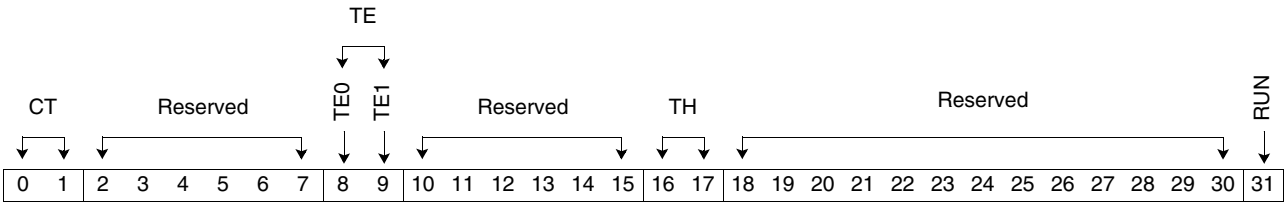
Notes:

The following information applies to the tables used at the beginning of each register description in this section.

1. In this section, power-on reset (POR) is defined as the sequence that starts when power is first applied to the chip and ends when the load function completes.
2. *Value at Initial POR* is the value that was initialized during the scan initialization or configuration ring part of the POR sequence.
3. Some fields within the architected registers are not physically implemented. Writing to these fields has no effect, and reading from these fields returns '0'. These fields are marked Reserved. The Rsvd_I bits are reserved and implemented. Writes to Rsvd_I bits are preserved on a read.

10.1.1 Control Register (CTRL)

Register Short Name	CTRL	**Privilege Type**	Read: Not privileged Write: Privileged
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	136 (Read) 152 (Write)	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan Initialization
Specification Type	PowerPC architected register	**Unit**	IU



Bit(s)	Field Name	Description
0:1	CT	Current thread active (Read Only) These read-only bits contain the current thread bits for threads 0 and 1. Software can read these bits to determine which thread they are operating on. Only one current thread bit is set at a time. 00 Reserved 01 Thread 1 is reading CTRL 10 Thread 0 is reading CTRL 11 Reserved
2:7	Reserved	Bits are not implemented; all bits read back zero.

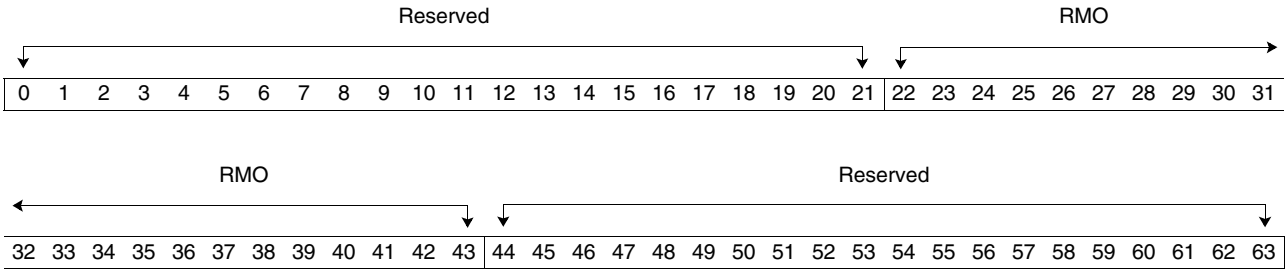
Bit(s)	Field Name	Description
8:9	TE	Thread enable bits (Read/Write) The hypervisor state can suspend its own thread by setting the TE bit for its thread to '0'. The hypervisor state can resume the opposite thread by setting the TE bit for the opposite thread to '1'. The hypervisor state cannot suspend the opposite thread by setting the TE bit for the opposite thread to '0'. This setting is ignored and does not cause an error. TE0 is the thread enable bit for thread 0. TE1is the thread enable bit for thread 1. If thread 0 executes the **mtctrl** instruction: [TE0, TE1] Description 00 Disable or suspend thread 0; thread 1 is unchanged. 01 Disable or suspend thread 0; enable or resume thread 1 if it was disabled. 10 Unchanged 11 Enable or resume thread 1 if it was disabled. If thread 1 executes the **mtctrl** instruction: [TE0, TE1] Description 00 Thread 0 is unchanged; disable or suspend thread 1. 01 Unchanged 10 Enable or resume thread 0 if it was disabled; disabled or suspend thread 1. 11 Enable or resume thread 0 if it was disabled. **Note:** Software should not disable a thread when in trace mode (MSR[SE] or MSR[BE] set to '1'). Doing so causes SRR0 to be undefined and can cause a system livelock hang condition.
10:15	Reserved	Bits are not implemented; all bits read back zero.
16:17	TH	Thread history If thread A writes CTRL[RUN], then CTRL[16] is set; otherwise, if thread B writes CTRL[31], then CTRL[17] is set. These bits cannot be set directly by writing bits [16] or [17] with a **mtctrl** instruction. They are only set when a thread writes CTRL[RUN].
18:30	Reserved	Bits are not implemented; all bits read back zero.
31	RUN	Run state bit.

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

10.1.2 Real Mode Offset Register (RMOR)

Register Short Name	RMOR	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	312	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	PowerPC architected register	**Unit**	MMU



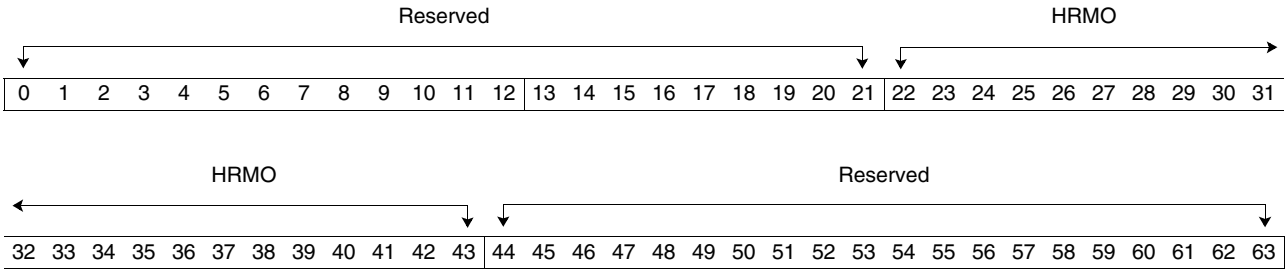
Bit(s)	Field Name	Description
0:21	Reserved	Bits are not implemented; all bits read back zero.
22:43	RMO	Real mode offset The offset from x‘0’ at which real-mode memory begins.
44:63	Reserved	Bits are not implemented; all bits read back zero.

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

10.1.3 Hypervisor Real Mode Offset Register (HRMOR)

Register Short Name	HRMOR	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	313	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	PowerPC architected register	**Unit**	MMU



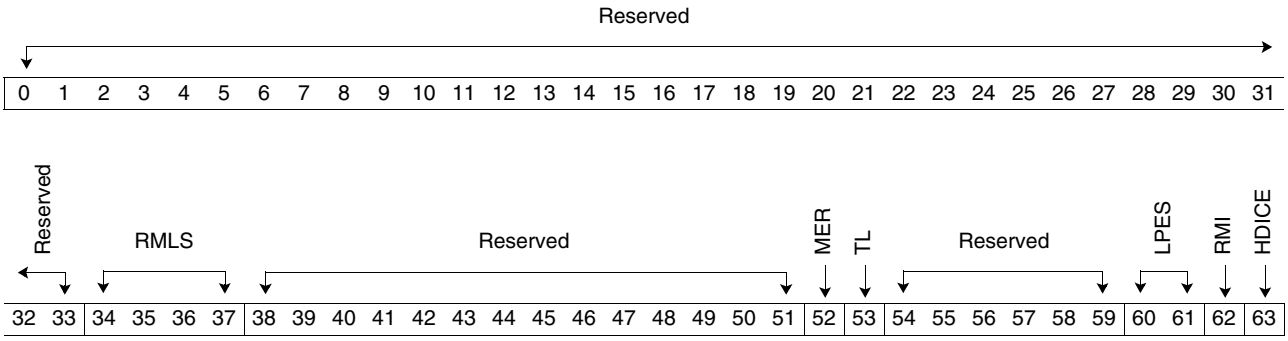
Bit(s)	Field Name	Description
0:21	Reserved	Bits are not implemented; all bits read back zero.
22:43	HRMO	Hypervisor real mode offset The offset from x'0' at which hypervisor real-mode memory begins.
44:63	Reserved	Bits are not implemented; all bits read back zero.

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

10.1.4 Logical Partition Control Register (LPCR)

Register Short Name	LPCR	Privilege Type	Hypervisor
Access Type	SPR Read/Write	Width	64 bits
Decimal SPR Number	318	Register Duplicated for Multithreading?	Partially (See notes in the bit definitions below)
Value at Initial POR	All bits set to zero	Value During POR Set By	Scan initialization during POR
Specification Type	PowerPC architected register	Unit	MMU



Bit(s)	Field Name	Description
0:33	Reserved	Bits are not implemented; all bits read back zero.
34:37	RMLS	Real mode limit selector This field is shared by both threads.
38:51	Reserved	Bits are not implemented; all bits read back zero.
52	MER	Mediate external exception request (interrupt enable) This field is duplicated per thread.
53	TL	Translation Lookaside Buffer (TLB) load This field is shared by both threads. 0 TLB loaded by processor 1 TLB loaded by hardware
54:59	Reserved	Bits are not implemented; all bits read back zero.
60:61	LPES	Logical partitioning (environment selector) This field is shared by both threads.
62	RMI	Real-mode caching (caching inhibited) This field is duplicated per thread.
63	HDICE	Hypervisor decrementer interrupt control enable This field is duplicated per thread.

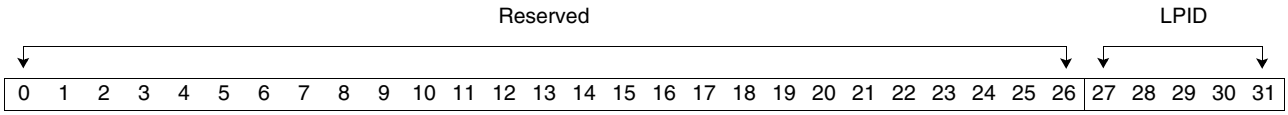
Programming Note: Mediated External Interrupts are not yet defined in *PowerPC Operating Environment Architecture, Book III*. This facility is a proposed extension and is expected to be made part of the architecture.

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

10.1.5 Logical Partition Identity Register (LPIDR)

Register Short Name	LPIDR	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	319	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	PowerPC architected register	**Unit**	MMU



Bit(s)	Field Name	Description
0:26	Reserved	Bits are not implemented; all bits read back zero.
27:31	LPID	Logical Partition ID

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

Programming Note:

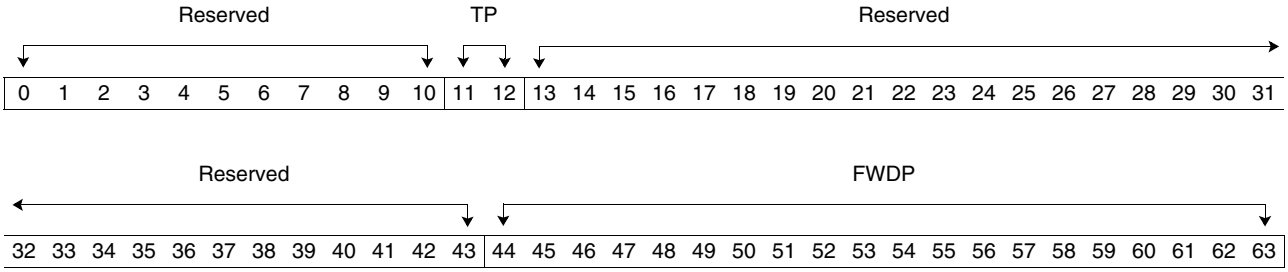
- TLB entries are tagged with the LPID when they are created. Therefore, the TLB does not need to be invalidated on a partition context switch.

10.1.6 Thread Status Register Local (TSRL)

This register allows a thread to read its own status.

Each thread has a Thread Status Register (TSR). When a thread reads its own TSR register, this register is called the Thread Status Register Local (TSRL). When a thread reads the TSR for the other thread, this register is called the Thread Status Register Remote (TSRR).

Register Short Name	TSRL	**Privilege Type**	Not Privileged
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	896	**Register Duplicated for Multithreading?**	Yes
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IU



Bit(s)	Field Name	Description
0:10	Reserved	Bits are not implemented; all bits read back zero.
11:12	TP	Thread priority (read/write) 00 Disabled 01 Low 10 Medium 11 High. If a System Reset interrupt is taken, this field is set to ‘11’. A thread cannot disable itself by attempting to directly set the TP field to ‘00’ (an attempt to do so is ignored). The thread must be disabled by setting the CTRL register appropriately. The Thread Status Control Register (TSCR) controls which thread priorities can be selected. When in problem state (MSR[PR] = ‘1’): • If TSCR[UCP] is set to ‘0’, the priority cannot be changed. • If TSCR[UCP] is set to ‘1’, the priority can be set to low or medium. When in privileged but nonhypervisor state (MSR[HV,PR] = ‘00’): • If TSCR[UCP, PSCTP] is set to ‘00’, the priority cannot be changed. • If TSCR[UCP, PSCTP] is set to ‘10’, the priority can be set to low or medium. • If TSCR[PSCTP] is set to ‘1’, the priority can be set to low, medium, or high. When in hypervisor state (MSR[HV, PR] = ‘10’), the priority can be set to low, medium, or high. **Note:** Attempts to set the TP field illegally are ignored, and the priority does not change.
13:43	Reserved	Bits are not implemented; all bits read back zero.

Bit(s)	Field Name	Description
44:63	FWDP	Forward progress timer (Read Only) This field is loaded from TTR[TTIM] each time the current thread completes a PowerPC Architecture instruction. This resets the timer to the maximum count. If the current thread is not disabled (TSRL[TP] = ‘00’), this field is decremented by one each time an instruction completes on the opposite thread. If TSCR[FPCF] = ‘1’ and the timer reaches x‘00001’, then after the next instruction completes, instructions for the opposite thread are flushed and no dispatch slots are given to the opposite thread until one instruction completes on the current thread. This field stops decrementing at x‘00001’ (the minimum count). This field is Initialized at POR to x‘00000’ (the maximum count).

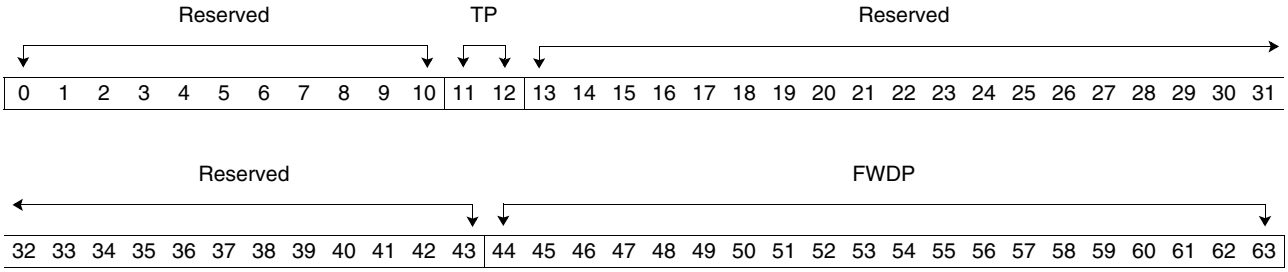
Related Registers: *Section 10.1.7 Thread Status Register Remote (TSRR)* on page 161

10.1.7 Thread Status Register Remote (TSRR)

This register allows a thread to read the status of the other thread.

Each thread has a Thread Status Register (TSR). When a thread reads its own TSR register, this register is called the Thread Status Register Local (TSRL). When a thread reads the TSR of the other thread, this register is called the Thread Status Register Remote (TSRR).

Register Short Name	TSRR	**Privilege Type**	Not Privileged
Access Type	SPR Read Only	**Width**	64 bits
Decimal SPR Number	897	**Register Duplicated for Multithreading?**	Yes
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IU

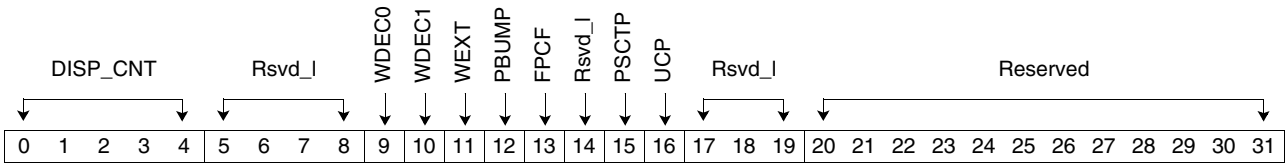


Bit(s)	Field Name	Description
0:10	Reserved	Bits are not implemented; all bits read back zero.
11:12	TP	Thread priority. Shows the thread priority of the opposite thread. 00 Disabled 01 Low priority 10 Medium priority 11 High priority
13:43	Reserved	Bits are not implemented; all bits read back zero.
44:63	FWDP	Forward progress timer Shows the Forward Progress Timer counter value of the opposite thread. See the TSRL[FWDP] field description.

Related Registers: *Section 10.1.6 Thread Status Register Local (TSRL)* on page 159

10.1.8 Thread Switch Control Register (TSCR)

Register Short Name	TSCR	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	921	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IU



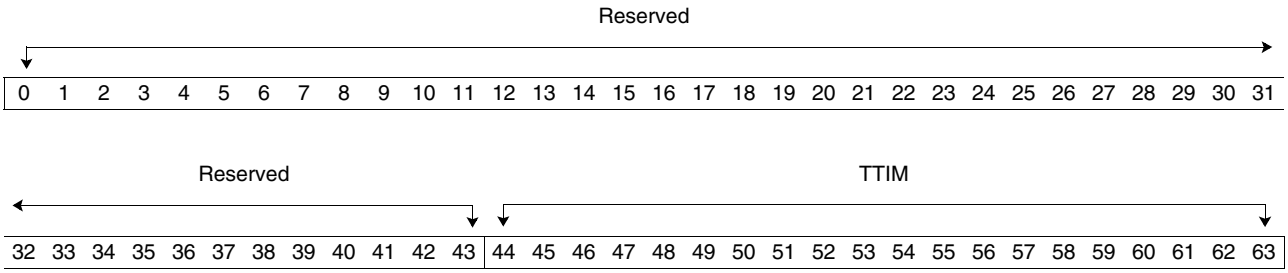
Bit(s)	Field Name	Description
0:4	DISP_CNT	Thread dispatch count Used to control the number of dispatch cycles each thread is given, based on the priority of the thread. A DISP_CNT of '00000' equals 32, which is the maximum dispatch count. During normal operation, this field should be set to 4.
5:8	Rsvd_l	Reserved. Latch bit is implemented; value read is the value written.
9	WDEC0	Decrementer wakeup enable for thread 0 0 Disabled 1 If a decrementer exception exists and the corresponding thread is suspended, then the thread is activated.
10	WDEC1	Decrementer wakeup enable for thread 1 0 Disabled 1 If a decrementer exception exists and the corresponding thread is suspended, then the thread is activated.
11	WEXT	External interrupt wakeup enable 0 Disabled 1 If an external interrupt exception exists and the corresponding thread is suspended, then the thread is activated.
12	PBUMP	Thread priority boost enable 0 Disabled 1 If a system-caused interrupt exception is presented, and the corresponding interrupt is not masked, and the priority of the corresponding thread is less than medium, sets the priority of the thread to medium. The hardware internally boosts the priority level to medium when the interrupt is pending. This does not change the value in the TSRL[TP] bits for the affected thread. The internal priority remains boosted to medium until an **mttsrl** or a priority-changing **nop** instruction occurs.
13	FPCF	Forward progress count flush enable **Note:** This bit only enables or disables the flush from occurring. The forward progress timer does not stop decrementing when set to zero. During normal operation, this bit should be set to '1'.
14	Reserved	Reserved. Latch bit is implemented; value read is the value written.

Bit(s)	Field Name	Description
15	PSCTP	Privileged but not hypervisor state change thread priority enable Enables the privileged state but not the hypervisor state (MSR[HV,PR] = '00') to change priority with "**or** Rx, Rx, Rx" **nop** instructions or with writes to TSRL[TP]. 0 The ability of the privileged state to change thread priority is determined by TSCR[UCP]. 1 The privileged state can change thread priority to low, medium, or high.
16	UCP	Problem state change thread priority enable Enables the problem state to change priority with "**or** Rx, Rx, Rx" **nop** instructions or writes to TSRL[TP]. 0 The problem state cannot change thread priority. 1 The problem state can change thread priority to low or medium only.
17:19	Rsvd_I	Reserved. Latch bit is implemented; value read is the value written.
20:31	Reserved	Bits are not implemented; all bits read back zero.

10.1.9 Thread Switch Timeout Register (TTR)

This register is used to ensure forward progress of the instruction dispatch.

Register Short Name	TTR	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	922	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IU



Bit(s)	Field Name	Description
0:43	Reserved	Bits are not implemented; all bits read back zero.
44:63	TTIM	Thread timeout flush value A value of x‘00000’ generates the maximum count. See the description of TSRL[FWDP]. The recommended value for this field is x‘04000’. This setting, along with the TSCR[FPCF] set to ‘1’, is necessary to guarantee that a forward progress timeout does not occur because one thread prevents the other from completing instructions. If one thread executes 16 K instructions without the other completing one, then the first thread is blocked even if it has higher priority. The second thread gets full resources to execute an instruction.

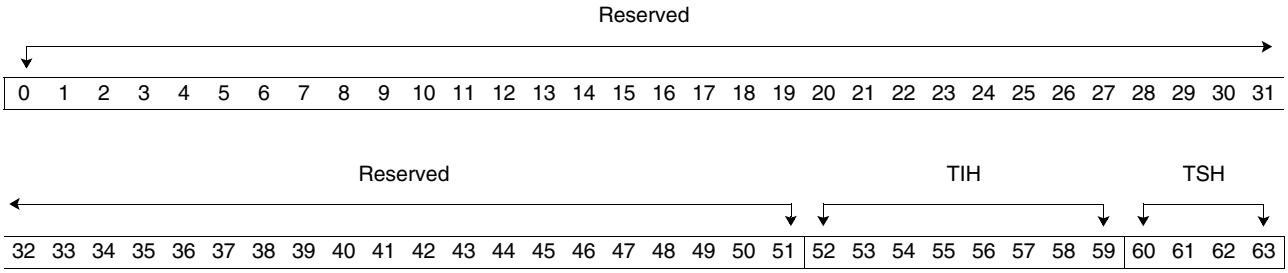
10.1.10 Translation Lookaside Buffer Special Purpose Registers

This section describes the translation lookaside buffer (TLB) special purpose registers (SPRs).

10.1.10.1 PPE Translation Lookaside Buffer Index Hint Register (PPE_TLB_Index_Hint)

Hardware updates this register when a TLB miss occurs with LPCR[TL] set to '1' (software tablewalk mode).

Register Short Name	PPE_TLB_Index_Hint	**Privilege Type**	Read: Privileged
Access Type	SPR Read Only	**Width**	64 bits
Decimal SPR Number	946	**Register Duplicated for Multithreading?**	Yes
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MMU



Bit(s)	Field Name	Description
0:51	Reserved	Bits are not implemented; all bits read back zero.
52:59	TIH	PPE TLB index hint The Index (congruence class) of the TLB that the hardware least recently used (LRU) facility would have chosen to replace if hardware TLB updates were enabled (LPCR[TL] = '0').
60:63	TSH	TLB set hint The recommended set for replacement (software replaces entries in the correct order to maintain the LRU). Valid values are: 1000 Set 0 0100 Set 1 0010 Set 2 0001 Set 3

Programming Note:

- The PPE_TLB_Index_Hint register is separate from the PPE_TLB_Index to avoid the possibility of hardware changing the index due to a fault while software is updating a TLB entry.
- This implementation supports a 256 × 4 TLB array. Hence, 8 fully-encoded bits [52:59] are used to choose among the 256 rows (or congruence classes) of the TLB, and 4 fully-decoded bits [60:63] are used to choose among the four columns (or sets).
- The effective address that caused the TLB miss that led to this register being set can be determined from the Data Address Register (DAR).

Related Registers: *Section 10.1.10.2 PPE Translation Lookaside Buffer Index Register (PPE_TLB_Index)* on page 167

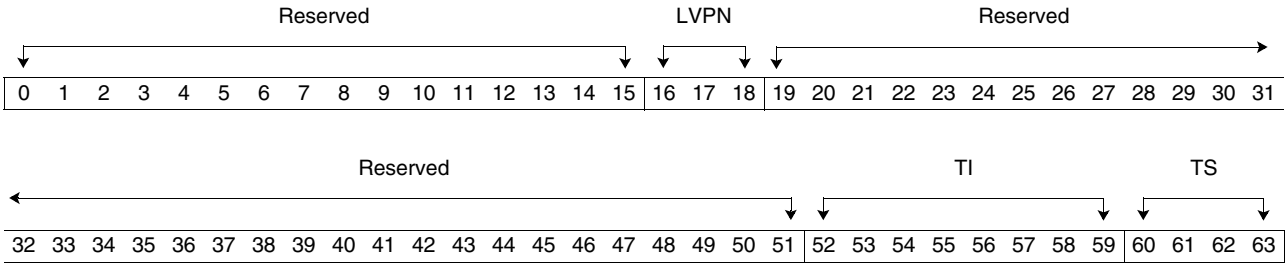
Additional Information:

- *Cell Broadband Engine Architecture*

10.1.10.2 PPE Translation Lookaside Buffer Index Register (PPE_TLB_Index)

This register is readable for diagnostic purposes only.

Register Short Name	PPE_TLB_Index	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	947	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MMU



Bit(s)	Field Name	Description
0:15	Reserved	Bits are not implemented; all bits read back zero.
16:18	LVPN	Lower virtual-page number LVPN[0:2] correspond to VPN[57:59]. **Note:** The abbreviated virtual-page number (AVPN)[0:56] corresponds to VPN[0:56]. The VPN corresponds to AVPN concatenated with LVPN. The PPU only implements LVPN[0:2], since LVPN[3:12 – *p*] is implied by the TI field of the PPE_TLB_Index register.
19:51	Reserved	Bits are not implemented; all bits read back zero.
52:59	TI	PPE TLB Index The fully-encoded index of the TLB chosen for replacement.
60:63	TS	TLB set The set chosen for replacement. The following are valid set combinations: 1000 Set 0 0100 Set 1 0010 Set 2 0001 Set 3 Setting multiple bits causes multiple sets in the TLB to be written with identical data. This is not recommended because it makes inefficient use of the TLB.

Programming Note:

- This implementation supports a 256 × 4 TLB array. Hence, 8 fully-encoded bits [52:59] are used to choose among the 256 rows (or congruence classes) of the TLB, and 4 fully-decoded bits [60:63] are used to choose among the four columns (or sets).
- This register is read by the memory management unit (MMU) hardware when an **mtspr** or **mfspr** instruction is executed with a target address of either the PPE_TLB_VPN or PPE_TLB_RPN. This tells the MMU which entry of the TLB software it should update. Software writes to this register to indicate the desired entry to replace. Subsequent writes to the PPE_TLB_VPN and PPE_TLB_RPN are based on the last written value to this register.

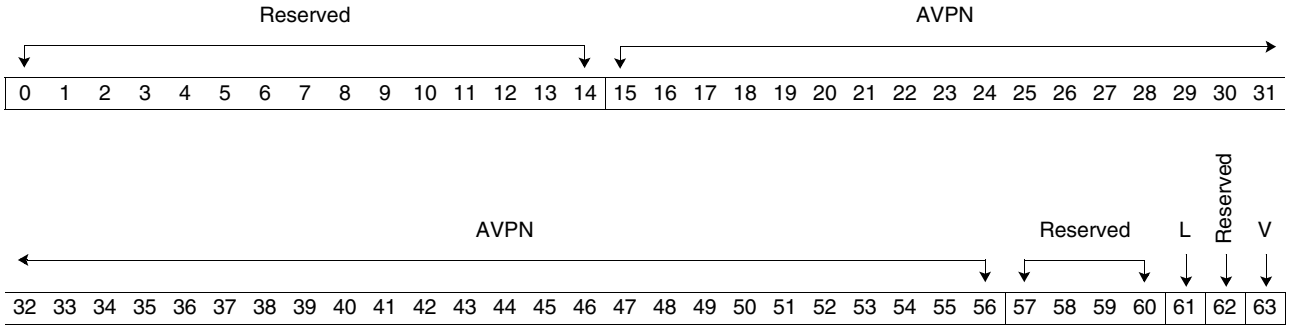
Related Registers: *Section 10.1.10.1 PPE Translation Lookaside Buffer Index Hint Register (PPE_TLB_Index_Hint)* on page 165

Additional Information:

- *Cell Broadband Engine Architecture*

10.1.10.3 PPE Translation Lookaside Buffer Virtual-Page Number Register (PPE_TLB_VPN)

Register Short Name	PPE_TLB_VPN	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	948	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MMU



Bit(s)	Field Name	Description
0:14	Reserved	Bits are not implemented; all bits read back zero.
15:56	AVPN	Abbreviated virtual-page number (AVPN) The AVPN corresponds to VPN[15:56]. For a description of the AVPN, see *PowerPC Operating Environment Architecture, Book III.* **Note:** When reading a 16-MB TLB entry, bit [56] of the AVPN is undefined, and software should ignore it.
57:60	Reserved	Bits are not implemented; all bits read back zero.
61	L	Large-page mode 0 4-KB page 1 Large page
62	Reserved	Bits are not implemented; all bits read back zero.
63	V	Valid bit 0 Invalid 1 Valid

Programming Note:

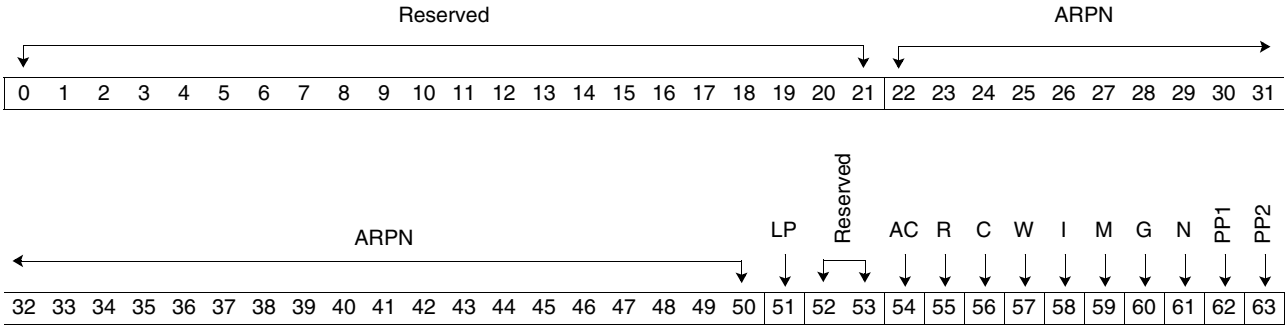
- If the VPN is being invalidated to change the protection attributes of a page or to steal the page, a TLB Invalidate Entry command must be issued to invalidate any cache of the effective-to-real address translation that may be associated with the TLB entry being invalidated.
- This register acts as a placeholder for the TLB entry pointed to by PPE_TLB_Index. Changing the value of PPE_TLB_Index implicitly changes the value of this register to match the contents of the corresponding TLB array entry pointed to by PPE_TLB_Index.
- This register is meant to be written as part of a sequence of instructions.

Additional Information:

- *Cell Broadband Engine Architecture*

10.1.10.4 PPE Translation Lookaside Buffer Real-Page Number Register (PPE_TLB_RPN)

Register Short Name	PPE_TLB_RPN	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	949	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MMU



Bit(s)	Field Name	Description
0:21	Reserved	Bits are not implemented; all bits read back zero.
22:50	ARPN	Abbreviated real page number The ARPN corresponds to RPN[22:50]. To obtain the full 30-bit RPN, the ARPN is combined with the LP field.
51	LP	Large page size selector If PPE_TLB_VPN[L] is set to '1', then bit [51] is used as the page size selector; otherwise, bit [51] corresponds to RPN[51].
52:53	Reserved	Bits are not implemented; all bits read back zero.
54	AC	Address compare
55	R	Reference This bit is always treated as '1' by the implementation. Any attempt to set it to '0' is ignored.
56	C	Change
57	W	Write-through This bit is forced to '0' in this implementation.
58	I	Caching inhibited
59	M	Memory coherency bit Memory is always coherent on this processor, so this value is forced to '1'. **Note:** Reference and change bit updates are done with M = '1'. Software should not set the page table entry (PTE) M bit to '0' because it may be implicitly overwritten by a reference or change bit update.
60	G	Guarded
61	N	No execute 0 Execute page 1 No-execute page
62	PP1	Page-protection bit 1 for tags inactive mode

Bit(s)	Field Name	Description
63	PP2	Page-protection bit 2 for tags inactive mode

Programming Note:

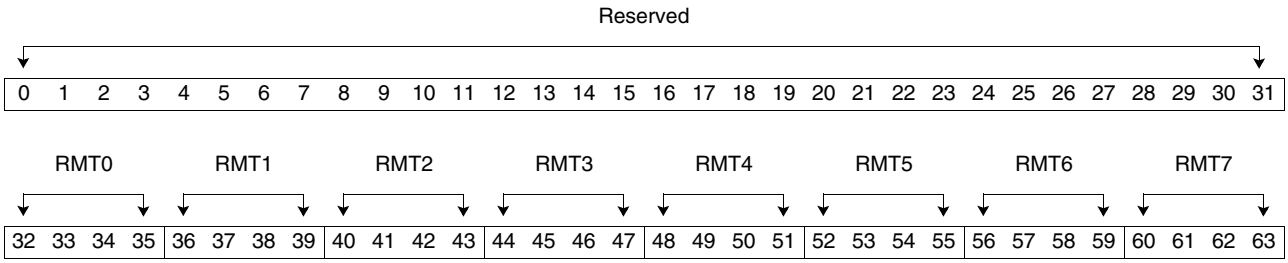
- This implementation supports two concurrent large page sizes. The selection between the two large pages sizes is controlled by the LP field when the L bit is set in the PPE_TLB_VPN register. The address of the real page (that is, the Real Page Number or RPN) is formed by concatenating the ARPN field with a zero when the L bit is set. Since the RPN must be on a page size boundary, some low-order bits must be set to zero by software when L is set or the results are undefined. For 64 KB, 1 MB, and 16 MB pages, the low order 4, 8, and 12 bits of the RPN respectively are zero. If the L bit is not set in the PPE_TLB_VPN, the RPN is formed by concatenating the ARPN with the LP field and the page size is 4 KB.
- This register acts as a placeholder for the TLB entry pointed to by PPE_TLB_Index. Changing the value of PPE_TLB_Index implicitly changes the value of this register to match the contents of the corresponding TLB array entry pointed to by PPE_TLB_Index.
- This register is meant to be written as part of a sequence of instructions.

Additional Information:

- *Cell Broadband Engine Architecture*

10.1.11 PPE Translation Lookaside Buffer RMT Register (PPE_TLB_RMT)

Register Short Name	PPE_TLB_RMT	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	951	**Register Duplicated for Multithreading?**	No
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	MMU



Bit(s)	Field Name	Description
0:31	Reserved	Bits are not implemented; all bits read back zero.
32:35	RMT0	Entry 0 of the replacement management table (RMT)
36:39	RMT1	Entry 1 of the RMT
40:43	RMT2	Entry 2 of the RMT
44:47	RMT3	Entry 3 of the RMT
48:51	RMT4	Entry 4 of the RMT
52:55	RMT5	Entry 5 of the RMT
56:59	RMT6	Entry 6 of the RMT
60:63	RMT7	Entry 7 of the RMT

Programming Note:

- Each RMT entry consists of 4 bits, which are fully decoded and correspond to a set in the TLB. If an effective address matches a range register, then the TLB considers the corresponding RMT entry for this range when replacing entries in the TLB. Each bit of the RMT entry can be thought of as a set enabler, indicating that, when set to '1', the corresponding set of the TLB is a valid entry to replace if the translation requires a TLB update to occur (for example, when the translation page table entry does not currently reside in the TLB). If multiple sets are indicated, they are replaced in priority order, beginning with invalid entries first, and then proceeding with valid entries from left to right.

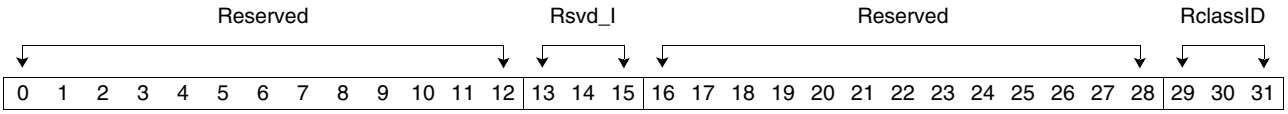
Additional Information:

- *Cell Broadband Engine Architecture*

10.1.12 Data Address Range SPRs

10.1.12.1 Data Class ID Register 0 (DCIDR0)

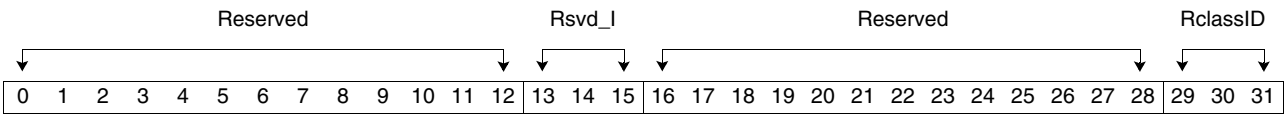
Register Short Name	DCIDR0	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	954	**Register Duplicated for Multithreading?**	Yes Each thread has a DCIDR0 and a DCIDR1.
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	XU



Bit(s)	Field Name	Description
0:12	Reserved	Bits are not implemented; all bits read back zero.
13:15	Rsvd_I	Reserved. Latch bit is implemented; value read is the value written.
16:28	Reserved	Bits are not implemented; all bits read back zero.
29:31	RclassID	Replacement class identifier (RclassID)

10.1.12.2 Data Class ID Register 1 (DCIDR1)

Register Short Name	DCIDR1	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	957	**Register Duplicated for Multithreading?**	Yes Each thread has a DCIDR0 and a DCIDR1.
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	XU

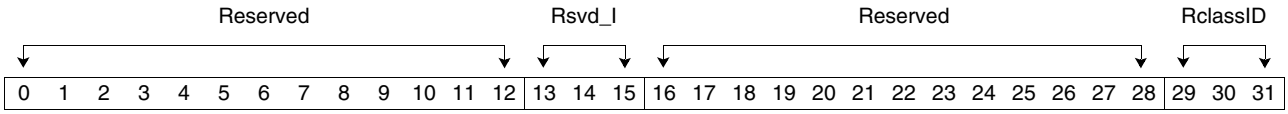


Bit(s)	Field Name	Description
0:12	Reserved	Bits are not implemented; all bits read back zero.
13:15	Rsvd_I	Reserved. Latch bit is implemented; value read is the value written.
16:28	Reserved	Bits are not implemented; all bits read back zero.
29:31	RclassID	RclassID

10.1.13 Instruction Range SPRs

10.1.13.1 Instruction Class ID Register 0 (ICIDR0)

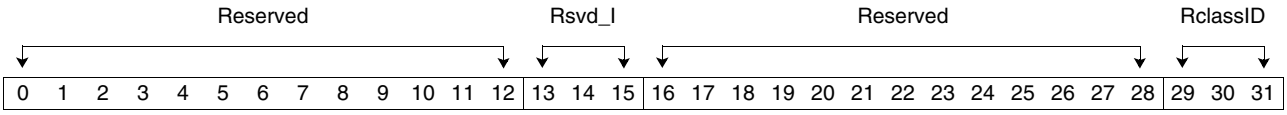
Register Short Name	ICIDR0	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	978	**Register Duplicated for Multithreading?**	Yes Each thread has an ICIDR0 and an ICIDR1.
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IU



Bit(s)	Field Name	Description
0:12	Reserved	Bits are not implemented; all bits read back zero.
13:15	Rsvd_I	Reserved. Latch bit is implemented; value read is the value written.
16:28	Reserved	Bits are not implemented; all bits read back zero.
29:31	RclassID	RclassID

10.1.13.2 Instruction Class ID Register 1 (ICIDR1)

Register Short Name	ICIDR1	**Privilege Type**	Hypervisor
Access Type	SPR Read/Write	**Width**	32 bits
Decimal SPR Number	981	**Register Duplicated for Multithreading?**	Yes Each thread has an ICIDR0 and an ICIDR1.
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	Implementation-specific register	**Unit**	IU



Bit(s)	Field Name	Description
0:12	Reserved	Bits are not implemented; all bits read back zero.
13:15	Rsvd_I	Reserved. Latch bit is implemented; value read is the value written.
16:28	Reserved	Bits are not implemented; all bits read back zero.
29:31	RclassID	RclassID

10.1.14 CBEA-Compliant Processor Version Register (BP_VR)

The BP_VR is a 32-bit read-only register that contains values that identify the version and revision level of the CBEA-compliant processor. The contents of the BP_VR are only accessible using a PPE move-from special-purpose register (**mfspr**) instruction. Read access to the BP_VR is privileged; write access is not provided. There is only one BP_VR per CBEA-compliant processor.

Version numbers are assigned by the CBEA process. Revision numbers are assigned by an implementation-defined process. The values currently assigned to these fields are as follows:

Design Level	Version	Revision
DD 1.0	x‘0000’	x‘0000’
DD 1.1	x‘0000’	x‘0001’
DD 2.0	x‘0000’	x‘0100’
DD 3.0	x‘0000’	x‘0200’
DD 3.1	x‘0000’	x‘0201’

Register Short Name	BP_VR	**Privilege Type**	Read: Privileged
Access Type	SPR Read only	**Width**	32
Decimal SPR Number	1022	**Register Duplicated for Multithreading?**	No
Value at Initial POR (for DD 3.1)	x‘00000000_00000201’	**Value During POR Set By**	Hardwired
Specification Type	CBEA architected register	**Unit**	XU

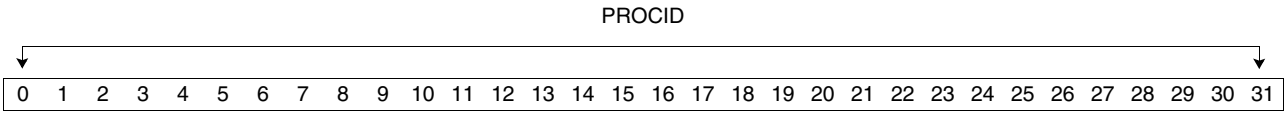
Related Registers: ***Appendix A.4.8 MFC Version Register (MFC_VR)*** on page 184 and *Appendix A.4.9 SPU Version Register (SPU_VR)*

Additional Information:

- For additional information, see *Cell Broadband Engine Architecture.*

10.1.15 Processor Identification Register (PIR)

Register Short Name	PIR	**Privilege Type**	Read: Privileged
Access Type	SPR Read only	**Width**	32 bits
Decimal SPR Number	1023	**Register Duplicated for Multithreading?**	Yes
Value at Initial POR	[0:22] 0 [23:30] Set by configuration chain [31] 0 for thread 0 1 for thread 1	**Value During POR Set By**	Scan initialization during POR Configuration ring
Specification Type	PowerPC architected register	**Unit**	XU



Bit(s)	Field Name	Description
0:31	PROCID	Processor ID

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

10.2 Special Architected Registers

This section describes registers that have been architected to meet special requirements. These registers may have unique characteristics, uses, and applications.

10.2.1 Machine State Register (MSR)

Register Short Name	MSR	**Privilege Type**	Privileged
Access Type	SPR Read/Write	**Width**	64 bits
Decimal SPR Number	N/A	**Register Duplicated for Multithreading?**	Yes
Value at Initial POR	All bits set to zero	**Value During POR Set By**	Scan initialization during POR
Specification Type	PowerPC architected register	**Unit**	IU

Bit(s)	Field Name	Description
0	SF	64-bit mode 0 The processor is in 32-bit mode. 1 The processor is in 64-bit mode.
1	TA	Tags-active mode This mode is not supported. This bit is reserved and is forced to zero.
2	Reserved	Bits are not implemented; all bits read back zero.
3	HV	Hypervisor State 0 The processor is not in hypervisor state. 1 If MSR[PR] = '0', then the processor is in hypervisor state; otherwise, the processor is in problem state.
4:37	Reserved	Bits are not implemented; all bits read back zero.
38	VXU	Vector/SIMD Multimedia Extension Unit (VXU) 0 The processor cannot execute VXU instructions. If the processor attempts to execute a VXU instruction, this causes a VXU Unavailable interrupt. 1 The processor can execute VXU instructions.
39:46	Reserved	Bits are not implemented; all bits read back zero.
47	ILE	Interrupt little-endian mode This mode is not supported. This bit is reserved and forced to '0'.

Bit(s)	Field Name	Description
48	EE	External interrupt enable 0 External and Decrementer interrupts are disabled. 1 External and Decrementer interrupts are enabled.
49	PR	Problem state 0 The processor is in privileged state. 1 The processor is in problem state.
50	FP	Floating-point available 0 The processor cannot execute any floating-point instructions. 1 The processor can execute floating-point instructions.
51	ME	Machine check enable 0 Machine Check interrupts are disabled. 1 Machine Check interrupts are enabled. **Note:** This bit is a hypervisor resource; see *PowerPC Operating Environment Architecture, Book III*.
52	FE0	Floating-point exception 0 The PPE does not support imprecise-nonrecoverable mode or imprecise-recoverable mode. The Floating-Point Exception Mode bits, FE0 and FE1, are interpreted as shown below: [FE0] [FE1] Mode 0 0 Ignore exceptions mode 0 1 Precise mode 1 0 Precise mode 1 1 Precise mode
53	SE	Single-step trace enable 0 The processor executes instructions normally. 1 The processor generates a single-step type of trace interrupt after successfully completing the execution of the next instruction (unless that instruction is **rfid**, **hrfid**, **attn**, or **sc**, which are never traced). Successful completion signifies that the instruction caused no other interrupt.
54	BE	Branch trace enable 0 The processor executes branch instructions normally. 1 The processor generates a branch type of trace interrupt after completing the execution of a branch instruction, whether or not the branch is taken.
55	FE1	Floating-point exception 1 **Note:** See the description of the FE0 bit in this register
56	Rsvd_l	Reserved. Latch bit is implemented; value read is the value written.
57	Reserved	Bits are not implemented; all bits read back zero.
58	IR	Instruction relocate 0 Instruction address translation is off. 1 Instruction address translation is on.
59	DR	Data relocate 0 Data address translation is off. 1 Data address translation is on.
60	Reserved	Bits are not implemented; all bits read back zero.
61	PMM	Performance monitor mark This bit is reserved and forced to zero.
62	RI	Recoverable interrupt 0 Interrupt is not recoverable. 1 Interrupt is recoverable.

Bit(s)	Field Name	Description
63	LE	Little-endian mode This mode is not supported. This bit is reserved and forced to '0', which means the mode is always big-endian.

Programming Note:

- A system reset sets the POR value of this register to the value indicated by the System Reset interrupt. This value is x'9000_0000_0000_0000' (where the 9 means MSR[0] is set to '1' and MSR[3] is set to '1').

Additional Information:

- *PowerPC Operating Environment Architecture, Book III*

Appendix A. Registers Defined in the CBEA

The tables below list the CBE registers whose implementation is identical to the architecture.

A.1 SPE Privilege 1 Registers

Table A-1. SPE Privilege 1 Registers

Register Name and (Short Name)	Additional Information
Class 1 Interrupt Mask Register (INT_Mask_class1)	See *Cell Broadband Engine Architecture*
Class 1 Interrupt Status Register (INT_Stat_class1)	See *Cell Broadband Engine Architecture*
MFC Address Compare Control Register (MFC_ACCR)	*Section A.4.1* on page 182
MFC Atomic Flush Register (MFC_Atomic_Flush)	*Section A.4.2* on page 182
MFC Data Address Register (MFC_DAR)	*Section A.4.3* on page 182
MFC Data-Storage Interrupt Status Register (MFC_DSISR)	*Section A.4.4* on page 182
MFC Real Mode Address Boundary Register (MFC_RMAB)	*Section A.4.5* on page 183
MFC State Register 1 (MFC_SR1)	*Section A.4.6* on page 183
MFC TLB Invalidate All Register (MFC_TLB_Invalidate_All)	Not implemented
MFC TLB Replacement Management Table Index Register (MFC_TLB_RMT_Index)	Not implemented. See *Section A.4.7* on page 183.
MFC Version Register (MFC_VR)	*Section A.4.8* on page 184
SPU Version Register (SPU_VR)	*Section A.4.9* on page 184

A.2 SPE Privilege 2 Registers

Table A-2. SPE Privilege 2 Registers

Register Name and (Short Name)	Additional Information
MFC Control Register (MFC_CNTL)	*Section A.5.1* on page 185
SLB Effective Segment ID Register (SLB_ESID)	*Section A.5.2* on page 185
SLB Invalidate All Register (SLB_Invalidate_All)	*Section A.5.3* on page 185
SPU Channel Data Register (SPU_ChnlData)	*Section A.5.4* on page 185
SPU Channel Index Register (SPU_ChnlIndex)	*Section A.5.5* on page 186
SPU Configuration Register (SPU_Cfg)	*Section A.5.6* on page 186
SPU Outbound Interrupt Mailbox Register (SPU_Out_Intr_Mbox)	*Section A.5.7* on page 186
SPU Privileged Control Register (SPU_PrivCntl)	*Section A.5.8* on page 186

A.3 SPE Problem State Registers

Table A-3. SPE Problem State Registers

Register Name and (Short Name)	Additional Information
MFC Command Tag Register (MFC_Tag)	*Section A.6.1* on page 187
MFC Effective Address Low Register (MFC_EAL)	*Section A.6.2* on page 187
MFC Transfer Size Register (MFC_Size)	*Section A.6.3* on page 187
Proxy Tag-Group Query Mask Register (Prxy_QueryMask)	*Section A.6.4* on page 187
Proxy Tag-Group Query Type Register (Prxy_QueryType)	*Section A.6.5* on page 187
Proxy Tag-Group Status Register (Prxy_TagStatus)	*Section A.6.6* on page 187
SPU Signal Notification Register 1 (SPU_Sig_Notify_1)	*Section A.6.7* on page 187
SPU Signal Notification Register 2 (SPU_Sig_Notify_2)	*Section A.6.8* on page 188
SPU Outbound Mailbox Register (SPU_Out_Mbox)	*Section A.7.1* on page 189
SPU Inbound Mailbox Register (SPU_In_Mbox)	*Section A.7.2* on page 189

A.4 SPE Privilege 1 Registers

A.4.1 MFC Address Compare Control Register (MFC_ACCR)

The MFC_ACCR allows the detection of DMA access to a virtual page marked with the address compare (AC) bit in the page-table entry (PTE) set and a range within the local storage.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.4.2 MFC Atomic Flush Register (MFC_Atomic_Flush)

The MFC_Atomic_Flush register is implementation dependent, and access is privileged. Privileged software uses this register to clear the contents of the cache used for atomic DMA commands and TLB-update operations. Data in the cache that is considered modified is pushed to memory, and the line is invalidated. Valid lines in the cache that are not considered modified are invalidated. The reservation is reset. For this operation to work properly, privileged software must suspend all MFC operations.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.4.3 MFC Data Address Register (MFC_DAR)

The MFC_DAR contains the 64-bit effective address (EA) from the DMA requests.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.4.4 MFC Data-Storage Interrupt Status Register (MFC_DSISR)

The MFC_DSISR register contains status bits relating to the data-storage interrupts (DSI) generated by the SMM.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.4.5 MFC Real Mode Address Boundary Register (MFC_RMAB)

The MFC supports a four-bit real-mode boundary (RMB) field. See the *Cell Broadband Engine Architecture* for more information about this register.

A.4.6 MFC State Register 1 (MFC_SR1)

The MFC_SR1 register contains configuration information controlled by a hypervisor. Access to this register is privileged. This register corresponds to the PPE Machine State Register (MSR).

In this implementation, MFC_SR1[58], the S bit, SPU Master Run Control, functions as an SPU-enable control. This bit allows privileged code the capability to temporarily suspend SPU execution and resume it again without affecting the SPU state. If the SPU is in the stopped state and MFC_SR1[58] is disabled (set to '0'), then the SPU stays in the stopped state. That is, status is unchanged from the last stopped reason code in the *SPU Status Register (SPU_Status)*. If MFC_SR1[58] is written with a start command, the SPU does not start and the SPU status remains unchanged; the last stopped reason code remains the same from the last stopped condition. However, reading the *SPU Run Control Register (SPU_RunCntl)* shows that the run command was accepted. If MFC_SR1[58] is then enabled (set to '1'), the SPU remains stopped. When the SPU_RunCntl register is next written with a run command, the SPU starts execution and the status is updated accordingly. See the *Cell Broadband Engine Architecture* for more information about this register.

A.4.7 MFC TLB Replacement Management Table Index Register (MFC_TLB_RMT_Index)

This register is not implemented in this version of the CBE.

Implementation Note: Only one translation fault may be outstanding. The implementation can either stop all command queue processing on the first translation error or continue processing. If processing is continued, all ordering rules must be followed (a command must not be processed if it is dependent on a command that is waiting for a translation fault to be resolved). The state of the MFC must appear as if the command (or partial command) were never issued. This is also the case if a second translation fault occurs.

A.4.8 MFC Version Register (MFC_VR)

The MFC_VR register contains a 32-bit value that identifies the specific version (model) and revision level of the MFC portion of the Cell Broadband Engine Architecture. The contents of this register are accessible from PPE using a load doubleword (**ld**) instruction. Read access to the MFC_VR register from PPE is privileged, and write access is not provided. Access to this register from the MFC is not provided. There is one MFC_VR register for each MFC in a Cell Broadband Engine.

Version numbers are assigned by the MFC architecture process. Revision numbers are assigned by an implementation-defined process.

The MFC_VR register distinguishes between processors that differ in attributes that may affect software. It contains two fields: Version and Revision. The value for this register for DD 1.0 was x‘000000000_000000000’, and the value for DD 1.1 was x‘000000000_000000001’. The value for DD 2.0 was x‘000000000_000000100’. The value for for this register DD 3.0 is x‘000000000_000000200’. The value listed below is for DD 3.1.

See the *Cell Broadband Engine Architecture* for more information about this register.

Register Short Name	MFC_VR	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400018’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	x‘00000000_00000201’	**Value During POR Set By**	Hardwired
Specification Type	CBEA architected register	**Unit**	MFC

Related Registers: *Section 10.1.14 CBEA-Compliant Processor Version Register (BP_VR)* and *Appendix A.4.9 SPU Version Register (SPU_VR)*

A.4.9 SPU Version Register (SPU_VR)

The SPU_VR register contains a 32-bit value that identifies the specific version (model) and revision level of the SPU portion of the Cell Broadband Engine Architecture. The contents of this register are accessible from PPE using a load doubleword (**ld**) instruction. Read access to the SPU_VR register from PPE is privileged, and write access is not provided. Access to this register from the SPU is not provided. There is one SPU_VR register for each SPU in a Cell Broadband Engine.

Version numbers are assigned by the SPU architecture process; revision numbers are assigned by an implementation-defined process.

The SPU_VR register distinguishes between processors that differ in attributes that may affect software. It contains two fields: Version and Revision. The value for this register for DD 1.0 and DD 1.1 was x‘000000000_000000000’. The value for this register for DD 3.0 is x‘000000000_000000200’. The value listed below is for DD 3.1.

See the *Cell Broadband Engine Architecture* for more information about this register.

Register Short Name	SPU_VR	**Privilege Type**	Privilege 1
Access Type	MMIO Read Only	**Width**	64 bits
Hex Offset From BE_MMIO_Base	SPE*n*: x‘400020’ + (x‘02000’ x *n*)	**Memory Map Area**	SPE Privilege 1
Value at Initial POR	x‘00000000_00000201’	**Value During POR Set By**	Hardwired
Specification Type	CBEA architected register	**Unit**	MFC

Related Registers: *Section 10.1.14 CBEA-Compliant Processor Version Register (BP_VR)* and *Appendix A.4.8 MFC Version Register (MFC_VR)*

A.5 SPE Privilege 2 Registers

A.5.1 MFC Control Register (MFC_CNTL)

The restart DMA operation is only effective if the DMA unit is in normal DMA queue operational status.

See the *Cell Broadband Engine Architecture* for more information on this register.

A.5.2 SLB Effective Segment ID Register (SLB_ESID)

To properly load the array with data, the SLB requires a write sequence similar to the one for the TLB. First, the index must be written to specify the entry for loading. The VSID and ESID fields are written independently, unlike the TLB writes, but the SLB_VSID write should follow the index. The SLB_ESID data is written last because it contains the Valid bit, and the entry should not be valid until all data is loaded.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.5.3 SLB Invalidate All Register (SLB_Invalidate_All)

A write to SLB_Invalidate_All causes the Valid bit in all SLB entries to be set to ‘0’, making the entries invalid. The remaining fields of each entry are undefined.

Writes to this register can be either 64-bit or 32-bit operations. Any write to the least significant word causes an entry in the SLB to be invalidated.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.5.4 SPU Channel Data Register (SPU_ChnlData)

If the data fields in the register are less than 32 bits (such as the SPU_RdEventStat channel, the SPU_WrEventMask channel, and the MFC_RdListStallStat channel), then only those bits defined in the register are updated, and the remaining bits of write data are ignored.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.5.5 SPU Channel Index Register (SPU_ChnlIndex)

The SPU Channel Index Register selects which SPU channel is accessed using the SPU Channel Count Register or the SPU Channel Data Register. Access to this register is privileged. Setting this register to channels that are not accessible via channel count or channel data has no effect. Reads of the non-accessible channels cause the channel count or channel data to return zeros. Writes are ignored. See the *Cell Broadband Engine Architecture* for bit definitions.

Note: In isolation mode, this register is forced to the SPU_WrDec channel, and all writes are ignored. The SPU_WrDec channel cannot be accessed through the channel count or channel data registers, so writes to this register have no effect, and reads return zeros.

A.5.6 SPU Configuration Register (SPU_Cfg)

The SPU Configuration Register is used to read or set the configuration of the SPU Signal Notification registers (SPU_Sig_Notify_1 and SPU_Sig_Notify_2) in the SPUs.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.5.7 SPU Outbound Interrupt Mailbox Register (SPU_Out_Intr_Mbox)

The PPU reads the mailbox data from the SPU in the SPU_Out_Intr_Mbox Register. The SPU stores to this mailbox by writing to the SPU_WrOutIntrMbox channel (see the *Cell Broadband Engine Architecture* for more information about the SPU Write Outbound Interrupt Mailbox Channel). When the SPU writes to the SPU_WrOutIntrMbox channel, the SPU_WrOutIntrMbox channel counter decrements from 1 to 0. Reading this register causes the SPU_WrOutIntrMbox channel counter to increment to 1. While the count is 1, further reads to this register do not affect the count, and the read data is the last stored value in this register. The queue depth for this implementation is 1.

Note: For the SPU_WrOutIntrMbox channel, the count is 1 at POR, and software re-initializes the count to 1.

This register has the same behavior as the SPU_Out_Mbox except that the SPU_Out_Intr_Mbox register has a PPU Mailbox Interrupt that allows the SPU to notify the PPU when the SPU has written data to the PPU Mailbox. The PPU Mailbox interrupt is asserted when the SPU_WrOutIntrMbox channel count transitions from 1 to 0, and deasserted when the SPU_WrOutIntrMbox channel counter increments from 0 to 1. The interrupt is taken when it is asserted and enabled. See the *Class 2 Interrupt Mask Register (INT_Mask_class2)* for information about how to enable this interrupt.

See the *Cell Broadband Engine Architecture* for more information on this register.

Related Register: See the *SPU Outbound Mailbox Register (SPU_Out_Mbox)* on page 189.

A.5.8 SPU Privileged Control Register (SPU_PrivCntl)

The SPU Privileged Control Register provides privileged software with the ability to control the execution environment of the SPU. Access to this register is privileged.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.6 SPE Problem State Registers

A.6.1 MFC Command Tag Register (MFC_Tag)

MFC_Size is the upper half of the MMIO word and MFC_Tag is the lower half of the same word. This word is written using a single, 32-bit store instruction.

See the *Cell Broadband Engine Architecture* for more information about this register.

Programming Note: The architecture allows for a future increase in the MFC_Tag register, along with the Prxy_QueryMask and Prxy_TagStatus registers, to seven bits.

A.6.2 MFC Effective Address Low Register (MFC_EAL)

The validity of this parameter is checked asynchronous to the instruction stream. If a segment fault, mapping fault, or protection violation occurs, MFC data segment exception is generated. If the address is not aligned, MFC DMA alignment exception is generated.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.6.3 MFC Transfer Size Register (MFC_Size)

MFC_Size is the upper half of the MMIO word and MFC_Tag is the lower half of the same word. This word is written using a single, 32-bit store instruction.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.6.4 Proxy Tag-Group Query Mask Register (Prxy_QueryMask)

The Proxy Tag-Group Query Mask register selects the tag groups to be included in the query operation.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.6.5 Proxy Tag-Group Query Type Register (Prxy_QueryType)

The Proxy Tag-Group Query Type register is used by software to request that the MFC detect a tag-group completion condition.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.6.6 Proxy Tag-Group Status Register (Prxy_TagStatus)

The Proxy Tag-Group Status register contains the current status of the tag groups enabled in the Proxy Tag-Group Query-Mask register.

See the *Cell Broadband Engine Architecture* for more information about this register.

A.6.7 SPU Signal Notification Register 1 (SPU_Sig_Notify_1)

See the *Cell Broadband Engine Architecture* for more information.

A.6.8 SPU Signal Notification Register 2 (SPU_Sig_Notify_2)

See the *Cell Broadband Engine Architecture* for more information.

A.7 SPU Control Registers

A.7.1 SPU Outbound Mailbox Register (SPU_Out_Mbox)

Other processors or devices read the mailbox data from the SPU in the SPU_Out_Mbox register. The SPU sends data to this mailbox by writing to the SPU_WrOutMbox channel (see *SPU Write Outbound Mailbox Channel (SPU_WrOutMbox)* in the *Cell Broadband Engine Architecture* for more information). When the SPU writes to the SPU_WrOutMbox channel, the channel counter decrements from 1 to 0. Reading this register causes the SPU_WrOutMbox channel counter to increment to 1. While the channel count is 1, further reads to this register do not affect the count, and the read data is the last stored value in this register. The queue depth for this implementation is 1.

Note: For the SPU Write Outbound Mailbox channel (SPU_WrOutMbox), the count is 1 at POR, and software re-initializes the count to 1.

The SPU Outbound Interrupt Mailbox register has the same behavior as this register, except that SPU_OutIntrMbox has a Mailbox Interrupt that allows the SPU to notify the PPU when the SPU has written data to the SPU_Out_Mbox.

See the *Cell Broadband Engine Architecture f*or more information about this register.

A.7.2 SPU Inbound Mailbox Register (SPU_In_Mbox)

The PPE writes mailbox data to the SPU in the SPU_In_Mbox register. This register corresponds to the SPU_RdInMbox channel. If this register is full, then additional writes overwrite the last entry written. The channel count remains at 4.

The SPU Inbound Mailbox Threshold interrupt is used to notify the PPE when the SPU has read all of the SPU_In_Mbox data. The SPU Inbound Mailbox Threshold interrupt is asserted when the SPU_RdInMbox channel count transitions from 1 to 0 (SPU_In_Mbox empty), and deasserted when the SPU_RdInMbox channel counter increments from 0 to 1. This interrupt is almost always asserted. The interrupt is taken when it is asserted and enabled.

See the *Class 2 Interrupt Mask Register (INT_Mask_class2)* for information about how to enable this interrupt. See the *Cell Broadband Engine Architecture* for more information about the SPU Inbound Mailbox Threshold interrupt.

Programming Note: SPU_NPC is not valid for an illegal instruction.

Glossary

ABIST	Automatic built-in self test.
AC0	Address concentrator 0.
ACK	See acknowledgment.
acknowledgment	A transmission control character that is sent as an affirmative response to a data transmission.
ALU	See arithmetic logical unit.
architecture	A detailed specification of requirements for a processor or computer system. It does not specify details of how the processor or computer system must be implemented; instead it provides a template for a family of compatible implementations.
arithmetic logical unit	The part of the central processing unit (CPU) that performs arithmetic, comparative, and logical functions.
ATO	Atomic unit. Part of an SPE's MFC. It is used to synchronize with other processor units.
BED	Cell Broadband Engine distribution bus.
BEI	Cell Broadband Engine interface.
BHT	Branch history table.
BIC	Bus interface controller. Part of the Cell Broadband Engine interface (BEI) to I/O.
BIF	Cell Broadband Engine interface. The EIB's internal communication protocol. It supports coherent interconnection for to other Cell Broadband Engines and BIF-compliant I/O devices, such as memory subsystems, switches, and bridge chips. See IOIF.
big-endian	A byte-ordering method in memory where the address n of a word corresponds to the most-significant byte. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the most-significant byte. See little-endian.
BIU	Bus interface unit. Part of the PPE's interface to the EIB.
BMT	Bandwidth management table.
BRU	Branch unit.
cache	High-speed memory close to a processor. A cache usually contains recently-accessed data or instructions, but certain cache-control instructions can lock, evict, or otherwise modify the caching of data or instructions.
cache coherency	The need to ensure that multiple threads on multiple processors changing a single cache line do not create inconsistent versions of the cache line in the different caches.

caching inhibited	A memory update policy in which the cache is bypassed, and the load or store is performed to or from main memory. A page of storage is considered caching inhibited when the “I” bit has a value of ‘1’ in the page table. Data located in caching inhibited pages cannot be cached at any memory hierarchy that is not visible to all processors and devices in the system. Stores must update the memory hierarchy to a level that is visible to all processors and devices in the system.
castouts	Cache blocks that must be written to memory when a cache miss causes a cache block to be replaced.
CBE	Cell Broadband Engine.
CBEA	Cell Broadband Engine Architecture. The Cell Broadband Engine is one implementation of the Cell Broadband Engine Architecture.
CEC	Central electronics complex.
channel	Channels are unidirectional, function-specific registers or queues. They are the primary means of communication between an SPE’s SPU and its MFC, which in turn mediates communication with the PPE, other SPEs, and other devices. These other devices use MMIO registers in the destination SPE to transfer information on the channel interface of that destination SPE. Specific channels have read or write properties, and blocking or nonblocking properties. Software on the SPU uses channel commands to enqueue DMA commands, query DMA and processor status, perform MFC synchronization, access auxiliary resources such as the decrementer (timer), and perform interprocessor-communication via mailboxes and signal-notification. See also memory channel.
CIU	Core interface unit.
CL	The class-ID parameter in an MFC command.
CO	See castouts.
coherence	Refers to memory and cache coherence. The correct ordering of stores to a memory address, and the enforcement of any required cache write-backs during accesses to that memory address. Cache coherence is implemented by a hardware snoop (or inquire) method, which compares the memory addresses of a load request with all cached copies of the data at that address. If a cache contains a modified copy of the requested data, the modified data is written back to memory before the pending load request is serviced.
corner case	A situation in which a rare combination of factors creates an unusual occurrence or a situation that involves the occurrence of the extreme values or limits of several variables.
CR	Condition register.
CRC	Cyclic redundancy check.

cresp	combined response.
CSA	Context-save area.
CSRA	Context save and restore area.
CTR	Counter register.
DAR	Data address register.
data storage interrupt	An interrupt posted when a fault is encountered accessing storage or I/O space. A typical data storage interrupt is a page fault or protection violation.
dcbf	Data cache block flush instruction.
dcbst	Data cache block store instruction.
dcbt	Data cache block touch x form instruction.
dcbtst	Data cache block touch for store instruction.
dcbz	Data cache block zero instruction.
decrementer	A register that counts down each time an event occurs. Each SPU contains dedicated 32-bit decrementers for scheduling or performance monitoring, by the program or by the SPU itself.
denormal (number)	Any nonzero number that is smaller than the smallest normal number is 'denormal'. Denormal numbers fill the gap around zero in floating point arithmetic.
D-ERAT	Data effective-to-real-address translation.
DERR	Data error.
direct-mapped cache	A cache in which each main memory address can appear in only one location within the cache, operates more quickly when the memory request is a cache hit.
DMA	Direct memory access. A technique for using a special-purpose controller to generate the source and destination addresses for a memory or I/O transfer.
DMA command	A type of MFC command that transfers or controls the transfer of a memory location containing data or instructions.
DMA queue	A set of two queues for holding DMA-transfer commands. The SPE's queue has 16 entries. The PPE's queue has four entries (two plus an additional two for L2 cache) for SPE-requested DMA commands, and eight entries for PPE-requested DMA commands.
DMAC	Direct memory access controller. A controller that performs DMA transfers.
double precision	The specification that causes a floating-point value to be stored (internally) in the long format (two computer words).
DP	See double-precision.
DR	Data relocate.

DSI	Data storage interrupt.
DSISR	Data storage interrupt status register.
dual-issue	Issuing two instructions at once, under certain conditions.
EA	Effective address.
EAH	MFC effective address high.
EAL	MFC effective address low.
ECB	External control bus.
ECC	See error correction code.
effective address	An address generated or used by a program to reference memory. A memory-management unit translates an effective address (EA) to a virtual address (VA), which it then translates to a real address (RA) that accesses real (physical) memory. The maximum size of the effective-address space is 2^{64} bytes.
EIB	Element interconnect bus. The on-chip coherent bus that handles communication between the PPE, SPEs, memory, and I/O devices (or a second Cell Broadband Engine). The EIB is organized as four unidirectional data rings (two clockwise and two counterclockwise).
EIC	External interrupt controller
eieio	Enforce in-order execution of I/O transaction.
ERAT	Effective-to-real address translation, or a buffer or table that contains such translations, or a table entry that contains such a translation.
error correction code	A code appended to a data block that can detect and correct multiple bit errors within the block.
ESID	Effective segment ID.
exception	An error, unusual condition, or external signal that may alter a status bit and will cause a corresponding interrupt, if the interrupt is enabled. See interrupt.
fence	An option for a barrier ordering command that causes the processor to wait for completion of all MFC commands before starting any commands queued after the **fence** command. It does not apply to these immediate commands: **getllar**, **putllc** and **putlluc**.
fetch	Retrieving instructions from either the cache or main memory and placing them into the instruction queue.
FIFO	First in first out. Refers to one way elements in a queue are processed. It is analogous to “people standing in line.”
FIR	Fault isolation register.

fixed point	Describing a number that has a specific number of decimal places (as in currency) or none at all (integers).
FlexIO	Rambus processor bus interface.
FLIH	First-level interrupt handler.
floating point	A way of representing real numbers (that is, values with fractions or decimals) in 32 bits or 64 bits. Floating-point representation is useful to describe very small or very large numbers.
FP	Floating point.
FPR	Floating-point register.
FPU	Floating-point unit.
fres	Floating reciprocal estimate single A form instruction.
frsqte	Floating reciprocal square root estimate A form instruction.
FXU	Fixed-point execution unit in the SPU. The FXU computes word shifts and rotates (WS), fixed-point operations (FX), and shuffle operations (SH).
general purpose register	An explicitly addressable register that can be used for a variety of purposes (for example, as an accumulator or an index register).
getllar	Get lock line and reserve command.
GPR	See general-purpose register.
GRF	Growable array file.
guarded	Prevented from responding to speculative loads and instruction fetches. The operating system typically implements guarding, for example, on all I/O devices.
HDEC	Hypervisor decrementer.
HID	Hardware-implementation dependent.
hypervisor	A control (or virtualization) layer between hardware and the operating system. It allocates resources, reserves resources, and protects resources among (for example) sets of SPEs that may be running under different operating systems. The Cell Broadband Engine has three operating modes: user, supervisor, and hypervisor. The hypervisor performs a meta-supervisor role that allows multiple independent supervisors' software to run on the same hardware platform. For example, the hypervisor allows both a real-time operating system and a traditional operating system to run on a single PPE. The PPE can then operate a subset of the SPEs in the Cell Broadband Engine with the real-time operating system, while the other SPEs run under the traditional operating system.
IABR	Instruction address breakpoint register.
IBUF	Instruction dispatch buffer.

IC or ICache	See instruction cache.
ICBI	Instruction cache block invalidate.
ICBIQ	**icbi** queue.
ID	Instruction dispatch.
I-ERAT	Instruction effective-to-real-address translation.
IFAR	Instruction fetch address register.
IIC	Internal interrupt controller.
implementation	A particular processor that conforms to the architecture, but may differ from other architecture-compliant implementations for example in design, feature set, and implementation of optional features.
instruction cache	A cache for providing program instructions to the processor faster than they can be obtained from RAM.
INT	Interrupt.
interrupt	A change in machine state in response to an exception. See exception.
interrupt packet	Used to signal an interrupt, typically to a processor or to another interruptible device.
IOC	I/O controller.
IOIF	Cell Broadband Engine I/O Interface. The EIB's noncoherent protocol for interconnection to I/O devices. See BIF.
IPI	Interprocessor interrupt.
IR	Instruction relocate.
IS	Instruction issue.
ISA	Instruction set architecture.
ISEG	Instruction segment exception.
ISI	Instruction storage interrupt.
ISRC	Interrupt source.
IU	Instruction unit.
JTAG	Joint Test Action Group.
KB	Kilobyte.
L1	Level-1 cache memory. The closest cache to a processor, measured in access time.

L2	Level-2 cache memory. The second-closest cache to a processor, measured in access time. An L2 cache is typically larger than an L1 cache.
LA	An LS address of a DMA list. It is used as a parameter in an MFC command.
LBIST	Logic built-in self test.
ld	Load doubleword instruction.
ldarx	Load doubleword and reserve x-form instruction.
LEAL	List element effective address low.
least recently used	A policy for a caching algorithm that removes from the cache the item that has the longest elapsed time since its last access., An algorithm used to identify and make available the cache space that contains the data that was least recently used.
least-significant bit	The bit of least value in an address, register, data element, or instruction encoding.
least-significant byte	The byte of least value in an address, register, data element, or instruction encoding.
little-endian	A byte-ordering method in memory where the address *n* of a word corresponds to the least-significant byte. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the most-significant byte. See big-endian.
livelock	An endless loop in program execution.
lmw	Load multiple word instruction.
local store	The 256-KB local store (LS) associated with each SPE. It holds both instructions and data.
logical partitioning	A function of an operating system that enables the creation of logical partitions.
LPAR	See logical partitioning.
LPID	Logical-partition identity.
LR	Link register.
LRU	See least recently used.
LS	See local store.
LSA	Local Store Address. An address in the LS of an SPU, by which programs running in the SPU and DMA transfers managed by the MFC access the LS.
LSb	See least significant bit.
LSB	See least significant byte.
LSCSA	Local storage context save area.
LSU	Load/store unit.
lswi	Load string word immediate instruction.

lswx	Load string word indexed instruction.
LTS	List element transfer size.
lwarx	Load word and reserve x-form instruction.
mailbox	A queue in an SPE's MFC for exchanging 32-bit messages between the SPE and the PPE or other devices. Two mailboxes (the SPU Write Outbound Mailbox and SPU Write Outbound Interrupt Mailbox) are provided for sending messages from the SPE. One mailbox (the SPU Read Inbound Mailbox) is provided for sending messages to the SPE.
main storage	The effective-address (EA) space. It consists physically of real memory (whatever is external to the memory-interface controller, including both volatile and nonvolatile memory), SPU LSs, memory-mapped registers and arrays, memory-mapped I/O devices (all I/O is memory-mapped), and pages of virtual memory that reside on disk. It does not include caches or execution-unit register files. See local store.
mask	A pattern of bits used to accept or reject bit patterns in another set of data. Hardware interrupts are enabled and disabled by setting or clearing a string of bits, with each interrupt assigned a bit position in a mask register
MB	Megabyte.
MBL	MIC bus logic.
memory channel	An interface to external memory chips. The Cell Broadband Engine supports two Rambus Extreme Data Rate (XDR) memory channels.
memory coherency	An aspect of caching in which it is ensured that an accurate view of memory is provided to all devices that share system memory.
memory-mapped	Mapped into the Cell Broadband Engine's addressable-memory space. Registers, SPE local stores (LSs), I/O devices, and other readable or writable storage can be memory-mapped. Privileged software does the mapping.
MERSI	Cache coherency protocol: Modified, Exclusive, Recent, Shared, Invalid, plus Mu (Unsolicited Modified), and Tagged (T)
MFC	Memory flow controller. It is part of an SPE and provides two main functions: moves data via DMA between the SPE's local store (LS) and main storage, and synchronizes the SPU with the rest of the processing units in the system.
mfceieio	MFC enforce in-order execution of I/O command.
mfcsync	MFC synchronize command.
mfspr	Move from special-purpose register instruction.
MIC	Memory interface controller. The Cell Broadband Engine's MIC supports two memory channels.
MMIO	Memory-mapped input/output. See memory-mapped.

MMU	Memory management unit. A functional unit that translates between effective addresses (EAs) used by programs and real addresses (RAs) used by physical memory. The MMU also provides protection mechanisms and other functions.
most-significant bit	The highest-order bit in an address, registers, data element, or instruction encoding.
most-significant byte	The highest-order byte in an address, registers, data element, or instruction encoding.
MSb	See most significant bit.
MSB	See most significant byte.
MSR	Machine state register.
MT	See multithreading.
mtmsr	Move to machine state register instruction.
mtspr	Move to special-purpose register instruction.
multithreading	Simultaneous execution of more than one program thread. It is implemented by sharing one software process and set of execution resources but duplicating the architectural state (registers, program counter, flags, and so forth) of each thread.
NCU	Non-cacheable unit.
no-op	No-operation. A single-cycle operation that does not affect registers or generate bus activity.
page	A region in memory. The PowerPC Architecture defines a page as a 4-KB area of memory, aligned on a 4-KB boundary or a large page size which is implementation dependent.
page fault	A page fault is a condition that occurs when the processor attempts to access a memory location that resides within a page not currently resident in physical memory.
page table	A table that maps virtual addresses (VAs) to real addresses (RAs) and contains related protection parameters and other information about memory locations.
PG	Processor-unit group.
PLG	Physical layer group.
POR	Power-on reset
PowerPC	Of or relating to the PowerPC Architecture or the microprocessors that implement this architecture.
PowerPC Architecture	A computer architecture that is based on the third generation of RISC processors. The PowerPC architecture was developed jointly by Apple, Motorola, and IBM.

PPE	PowerPC Processor Element. The general-purpose processor in the Cell Broadband Engine. Consists of the PPU and the PPSS.
PPSS	PowerPC Processor Storage Subsystem. Part of the PPE. It operates at half the frequency of the PPU and includes an L2 cache and Bus Interface Unit (BIU).
PPU	PowerPC Processor Unit. The part of the PPE that includes execution units, memory-management unit, and L1 cache.
privileged mode	Also known as supervisor mode. The permission level of operating system instructions. The instructions are described in *PowerPC Architecture, Book III* and are required of software the accesses system-critical resources.
problem state	The permission level of user instructions. The instructions are described in *PowerPC Architecture, Books I and II* and are required of software that implements application programs.
PTE	Page table entry. See page table.
PTEG	Page table entry group.
putllc	Put lock line conditional on a reservation command.
putlluc	Put lock line unconditional command.
PVR	Processor version register.
QoS	Quality of service. This usually relates to a guarantee of minimum bandwidth for streaming applications.
quadword	A group of 16 contiguous locations starting at an address divisible by 16.
RA	Real address.
RAG	Resource allocation group.
RC	Read-and-claim state machine.
rchcnt	Read channel counter instruction.
rdch	Read from channel instruction.
real address	An address for physical storage, which includes physical memory, the PPE's L1 and L2 caches, and the SPE's local stores (LSs) if the operating system has mapped the LSs to the real-address space. The maximum size of the real-address space is 2^{42} bytes.
RESN	Returned envelope sequence number. In the BIC. Upon successful reception of an envelope, a positive acknowledgment is generated back to the transmitting chip.
RLM	Random logic macro.
RMT	Replacement management table.
RPN	Real page number.

RRAC	Redwood Rambus Access Cell (RRAC) physical link (PHY). This is an early term for FlexIO, now obsolete.
scarfing	The direct transfer of data to the PPE L2 cache.
sdcrf	SL1 data-cache range flush instruction.
sdcrst	SL1 data-cache range store instruction.
sdcrt	SL1 data-cache range touch instruction.
sdcrtst	SL1 data-cache range touch for store instruction.
sdcrz	SL1 data-cache range set to zero instruction.
SDR	Storage descriptor register.
SG	SPU group.
signal	Information sent on a signal-notification channel. These channels are inbound (to an SPE) registers. They can be used by the PPE or other processor to send information to an SPE. Each SPE has two 32-bit signal-notification registers, each of which has a corresponding memory-mapped I/O (MMIO) register into which the signal-notification data is written by the sending processor. Unlike mailboxes, they can be configured for either one-to-one or many-to-one signalling. These signals are unrelated to UNIX signals. See channel and mailbox.
SIMD	Single instruction, multiple data. Processing in which a single instruction operates on multiple data elements that make up a vector data-type. Also known as vector processing. This style of programming implements data-level parallelism.
SL1	A first-level cache for DMA transfers between local storage and main storage.
SLB	Segment lookaside buffer. It is used to map an effective address (EA) to a virtual address (VA).
slbia	SLB invalidate all instruction.
slbie	SLB invalidate entry instruction.
slbmfee	SLB move from entry ESID X-form instruction.
slbmfev	SLB move from entry VSID X-form instruction.
slbmte	SLB move to entry X-form instruction.
SMM	Synergistic memory management unit. It translates EAs to RAs in an SPU.
SMP	Symmetric multiprocessor.
sndsig	Send signal command.
sndsigb	Update signal-notification registers in an I/O device or another SPU with barrier command.

sndsigf	Update signal notification registers in an I/O device or another SPU with fence command.
snoop	To compare an address on a bus with a tag in a cache, in order to detect operations that violate memory coherency.
SPE	Synergistic processor element. Consists of a synergistic processor unit (SPU), a memory flow controller (MFC), and local store (LS).
SPR	Special-purpose register.
SPU	Synergistic processor unit. The part of an SPE that executes instructions from its local store (LS).
SRR0/SRR1	Save and restore register 0 and 1 instruction.
stdcx	Store double word conditional indexed instruction.
stmw	Store multiple word instruction.
stswi	Store string word immediate instruction.
stswx	Store string word indexed x-form instruction.
stwcx	Store word conditional x-form instruction.
supervisor mode	The privileged operation state of a processor. In supervisor mode, software, typically the operating system, can access all control registers and can access the supervisor memory space, among other privileged operations.
sync	Synchronize command.
synchronization	The order in which storage accesses are performed.
TAG	MFC command tag.
tag group	A group of DMA commands. Each DMA command is tagged with a 5-bit tag group identifier. Software can use this identifier to check or wait on the completion of all queued commands in one or more tag groups. All DMA commands except **getllar**, **putllc**, and **putlluc** are associated with a tag group.
TCU	Pervasive unit, used for test control logic.
thread	A sequence of instructions executed within the global context (shared memory space and other global resources) of a process that has created (spawned) the thread. Multiple threads (including multiple instances of the same sequence of instructions) can run simultaneously, if each thread has its own architectural state (registers, program counter, flags, and other program-visible state). Each SPE can support only a single thread at any one time. The multiple SPEs can simultaneously support multiple threads. The PPE supports two threads at any one time, without the need for software to create the threads. The PPE does this by duplicating architectural state.
time base	Chip-level time base, as defined in the PowerPC Architecture.

TKM	Token management unit. Part of the element interconnect bus (EIB) that software can program to regulate the rate at which particular devices are allowed to make EIB command requests.
TLB	Translation lookaside buffer. An on-chip cache that translates virtual addresses (VAs) to real addresses (RAs). A TLB caches page-table entries for the most recently accessed pages, thereby eliminating the necessity to access the page table from memory during load/store operations.
tlbie	Translation lookaside buffer invalidate entry instruction.
TS	The transfer-size parameter in an MFC command.
VA	See virtual address.
vector	An instruction operand containing a set of data elements packed into a one-dimensional array. The elements can be fixed-point or floating-point values. Most Vector/SIMD Multimedia Extension and SPU SIMD instructions operate on vector operands. Vectors are also called SIMD operands or packed operands.
Vector/SIMD Multimedia Extension	The SIMD instruction set of the PowerPC Architecture, supported on the PPE.
virtual address	An address to the virtual-memory space, which is much larger than the physical address space and includes pages stored on disk. It is translated from an effective address (EA) by a segmentation mechanism and used by the paging mechanism to obtain the real address (RA). The maximum size of the virtual-address space is 2^{65} bytes.
VMRF	Vector multimedia register file.
VPN	Virtual page number. The number of the page in virtual memory.
VS	Virtual storage.
VSID	Virtual segment ID.
VSU	Vector/scalar unit.
VXU	Vector/SIMD multimedia extension unit.
WIMG bits	Four bits in the page table, also called a page table entry, which control the processor's accesses to cache and main storage. “W” stands for write through, “I” for cache inhibit, “M” for memory coherence, and “G” for guarded storage.
word	Four bytes.
wrch	Write to channel instruction.
XDR	Rambus XDR DRAM memory technology.
XIO	A Rambus XDR extreme data rate I/O (XIO) memory channel.
XU	Execution unit.

Revision Log

Revision Date	Contents of Modification
Version 1.0 November 9, 2005	Initial release.